FORUM NOKIA

# Hildon 2.2
# Widget UI Specification

Version 1.0; September 18, 2009

## Licence

## Contents

1 **Introduction** 12
1.1 Widgets 12
1.2 List of contributors 12
2 **Hildon touch view** 13
2.1 Structure 13
2.2 Default event actions 14
2.3 Implementation 14
2.3.1 The API for stacking HildonStackableWindows 14
3 **UI modes** 15
3.1 Normal mode 15
3.2 Do Not Disturb flag 15
3.2.1 Implementation 16
3.3 Edit mode 16
3.3.1 Structure 16
3.3.2 Implementation 16
3.4 Single selection mode 17
3.4.1 Structure 17
3.4.2 Behaviour 17
3.4.3 Implementation 18
3.5 Multiple selection mode 18
3.5.1 Structure 18
3.5.2 Behaviour 19
3.5.3 Implementation 19
3.6 Full-screen mode 20
3.6.1 Special cases and legacy support 20
4 **Hildon Touch Pannable Area** 21
4.1 Behaviour 21
4.2 Widgets supported inside the pannable area 23
4.3 Implementation 23
5 **Hildon Touch List** 25
5.1 Behaviour 26
5.2 Implementation 26
5.2.1 Row headers implementation 27
6 **Hildon Touch Grid** 28
6.1 Behaviour 29
6.2 Implementation 29

6.2.1 Row Headers implementation ..... 29
**7 Toolbars ..... 30**
7.1 Hildon Touch Toolbar ..... 30
7.1.1 Implementation ..... 30
7.2 Hildon touch Edit Mode Toolbar ..... 30
7.2.1 Structure ..... 31
7.2.2 Implementation ..... 31
**8 Hildon Touch View Menu ..... 32**
8.1.1 Structure ..... 32
8.1.2 Behaviour ..... 32
8.1.3 Implementation ..... 33
**9 Hildon Touch Context Menu ..... 34**
9.1 Behaviour ..... 34
9.2 Implementation ..... 35
**10 Banners and notes ..... 36**
10.1 Hildon Touch Information banner ..... 36
10.1.1 Implementation ..... 37
10.2 Hildon Touch Information note ..... 37
10.2.1 Implementation ..... 37
10.3 Hildon Touch Confirmation note ..... 37
10.3.1 Implementation ..... 39
10.4 Hildon Touch Incoming event ..... 39
10.4.1 Implementation ..... 39
**11 Dialogues ..... 40**
11.1 Modality of dialogues ..... 40
11.2 Hildon Touch Dialogue ..... 40
11.2.1 Structure ..... 41
11.2.2 Implementation ..... 42
11.3 Hildon Touch Wizard ..... 42
11.3.1 Structure ..... 42
11.3.2 Implementation ..... 44
**12 Text and images ..... 45**
12.1 Basic text ..... 45
12.1.1 Implementation ..... 45
12.2 Basic image ..... 45
12.2.1 Implementation ..... 45
**13 Buttons ..... 47**
13.1 Hildon Touch Finger button, Thumb button ..... 47

13.1.1 Structure ..... 47
13.1.2 Implementation ..... 48
13.2 Aligning and theming complex button text ..... 48
13.2.1 Implementation ..... 49
13.3 Hildon Touch Checkbox ..... 49
13.3.1 Structure ..... 49
13.3.2 Implementation ..... 50
13.4 Hildon Touch Toggle button ..... 50
13.4.1 Structure ..... 50
13.4.2 Implementation ..... 51
**14 Data controllers ..... 52**
14.1 Hildon Touch Date ..... 52
14.1.1 Structure ..... 52
14.1.2 Implementation ..... 52
14.2 Hildon Touch Time ..... 53
14.2.1 Structure ..... 53
14.2.2 Implementation ..... 53
**15 Pickers ..... 54**
15.1.1 Implementation ..... 55
15.2 Hildon Touch List picker ..... 55
15.2.1 Structure ..... 56
15.2.2 Implementation ..... 56
15.3 Hildon Touch List picker with multiple selection ..... 56
15.3.1 Structure ..... 56
15.3.2 Implementation ..... 56
15.4 Hildon Touch List picker with entry ..... 57
15.4.1 Structure ..... 57
15.4.2 Implementation ..... 57
15.5 Hildon Touch Date picker ..... 57
15.5.1 Structure ..... 58
15.5.2 Implementation ..... 58
15.6 Hildon Touch Time picker ..... 58
15.6.1 Structure ..... 59
15.6.2 Implementation ..... 59
**16 Text editors ..... 60**
16.1 Common features ..... 60
16.2 Placeholder text ..... 60
16.3 Hildon Touch Text Entry ..... 61

16.3.1 Implementation ... 61
16.4 Hildon Touch Text Area ... 61
16.4.1 Implementation ... 62
**17 Progress indicators and progress controllers ... 63**
17.1 Hildon Touch progress indicator ... 63
17.1.1 Implementation ... 64
17.2 Hildon Touch Progress Note ... 64
17.2.1 Implementation ... 65
17.3 Hildon Touch Progress Bar ... 65
17.3.1 Structure ... 66
17.3.2 Implementation ... 66
17.4 Hildon Touch Seek Bar ... 66
17.4.1 Structure ... 67
17.4.2 Implementation ... 67
17.4.3 Implementation for custom negative-positive Seek Bar ... 67
**18 Other UI notes ... 68**
18.1 Icons used by widgets ... 68
18.2 Common dialogue button labels ... 69
18.3 Date format strings ... 70
18.4 Time format strings ... 72
18.5 Other platform UI banners and strings ... 74
**19 Deprecated widgets ... 75**
19.1 Quick Reference: Used / Deprecated ... 76
19.1.1 Hildon Widgets ... 76
19.2 Gtk Widgets ... 76

## Tables

Table 1: Default event actions ..... 14
Table 2: List/Grid actions for normal mode events ..... 15
Table 3: List/Grid actions for single selection mode events ..... 17
Table 4: List/Grid actions for multiple selection mode events ..... 19
Table 5: Pannable area event actions ..... 21
Table 6: Pannable area widgets ..... 23
Table 7: Touch View Menu actions ..... 33
Table 8: Touch Context Menu actions ..... 34
Table 9: Functionality of notes ..... 36
Table 10: Touch Dialogue actions ..... 41
Table 11: Touch Wizard UI strings ..... 44
Table 12: Touch Button actions ..... 47
Table 13: Touch Button properties ..... 47
Table 14: Touch Checkbox actions ..... 49
Table 15: Touch toggle button actions ..... 50
Table 16: Text editor widget actions ..... 60
Table 17: Text editor widget properties ..... 60
Table 18: Touch Progress Bar widget properties ..... 66
Table 19: Touch Seek Bar widget actions ..... 67
Table 20: Touch Seek Bar widget properties ..... 67
Table 21: Widget icons and their descriptions ..... 69
Table 22: Button labels ..... 70
Table 23: Date format strings ..... 70
Table 24: Time format strings ..... 73
Table 25: Other banners and strings ..... 74
Table 26: Hildon widgets ..... 76
Table 27: Gtk widgets ..... 77

## Figures

Figure 1: Hildon windows structure, Root View, and Sub View ....13
Figure 2: Edit mode for a view ....16
Figure 3: Single selection mode for a list ....17
Figure 4: Multiple selection mode for a list ....18
Figure 5: Pannable  Area widget with pan indicator ....21
Figure 6: Pannable Area widget interaction diagram ....22
Figure 7: Touch List ....25
Figure 8: Touch List with group titles....25
Figure 9: Touch List in the multiple selection mode and in a dialogue ....26
Figure 10: Grid displaying pannable content ....28
Figure 11: Grid with group titles ....28
Figure 12: Toolbar ....30
Figure 13: Edit Mode Toolbar....31
Figure 14: View menu....32
Figure 15: Context menu....34
Figure 16: Information banner ....36
Figure 17: Information note ....37
Figure 18: Confirmation note ....38
Figure 19: Confirmation note with system modal dialogue ....38
Figure 20: Incoming event ....39
Figure 21: Touch Dialogue....40
Figure 22: Touch Dialogue with system modal dialogue ....41
Figure 23: Touch wizard example ....42
Figure 24: Finger Button and Thumb Button ....47
Figure 25: Button with longer text ....48
Figure 26: Complex button text with single line and two columns....48
Figure 27: Complex button text with two lines....48
Figure 28: Checkbox states....49
Figure 29: Toggle button states....50
Figure 30: Touch Date....52
Figure 31: Touch Time ....53
Figure 32: Picker widget diagram....54
Figure 33: Picket values (data formats) ....55
Figure 34: Touch List picker ....55
Figure 35: Touch List picker with multiple selection ....56
Figure 36: Touch List picker with entry ....57

Figure 37: Touch Date picker ........ 58
Figure 38: Touch Time picker ........ 59
Figure 39: Touch text Entry with widget states ........ 61
Figure 40: Touch Text Area with focus and entered text ........ 62
Figure 41: Progress indication in a view ........ 63
Figure 42: Progress indication in a dialogue ........ 64
Figure 43: Progress indication inside content ........ 64
Figure 44: Touch Progress Note ........ 65
Figure 45: Touch Progress Note with system modal dialogue ........ 65
Figure 46: Touch Progress Bar ........ 66
Figure 47: Touch Seek Bar ........ 67
Figure 48: Custom Seek bar ........ 67

## Examples

Example 1: Single selection mode implementation........18
Example 2: Multiple selection mode implementation........20
Example 3: Tappable image shadow implementation........46
Example 4: Placeholder text, entry asking phone number........61
Example 5: Date format strings........70

## Change history

| | | |
|---|---|---|
| September 18, 2009 | Version 1.0 | Initial document release |
| | | |

# 1 Introduction

This document specifies the Hildon 2.2 widgets, following the Hildon 2.2 UI style. For details of the UI Style, see the *Hildon 2.2 UI Style Guide*.

The display texts in application specifications are draft versions. The illustrations in this document do not refer to any real product and are designed to communicate the user interaction, not the display the exact graphics. The exact layout and graphics are defined in the *Fremantle Master Layout Guide*.

Widgets are described in Chapters 3-17.

The logical names and UI strings are specified in tables. Any other logical name or UI string explained in text should not be referred to.

## 1.1 Widgets

The widgets described here are an effort to refine the Hildon 2.1 toolkit to work better with fingers. Therefore, all dimensions and sizes are described in relation to the physical size on screen, which is then translated to pixels for given hardware. Thus, when adapting the content for another form factor or screen DPI, please note that in most cases the physical size is what matters, not pixel dimensions.

Some of the widgets are a new development in Hildon 2.2, while some others are just themed larger to adapt them for finger use. The rest of the widgets remain as is from Hildon 2.1. They are available in the toolkit, but not adapted for finger use. See http://maemo.org/api_refs/4.0/hildon/ for more information.

As described in *Hildon 2.2 UI Style Guide*, there is no initial focus. This means general HW key navigation is *not* supported in the UI. Only the cursor can be moved inside a single text input field with the HW arrow keys.

The UI Framework does not provide platform-wide support for portrait mode. There is no support for widgets in portrait mode.

## 1.2 List of contributors

The following people contributed in creating the current and/or the past versions of this document:

Kim Bergman, Patrik Flykt, Carlos Guerreiro, Jere Heikkinen, Teemu Heinonen, Antti Ijäs, Jussi-Pekka Kekki, Janne Korsumäki, Matti Kilponen, Risto Kivilahti, Tommi Komulainen, Tuomas Kuosmanen, Katja Kääriä, Karl Lattimer, Tommi Leino, Henri Melaanvuo, Camilla Mitts, Tuomo Mäkinen, Rodrigo Novo, Sampo Nurmentaus, Ville Paukkonen, Yannick Pellet, Antti Pelomaa, Hannu Pirskanen, Juho Paasonen, Teemu Pohjola, Sergei Pronin, Roope Rainisto, Visa Rauta, Jenni Romppainen, Pertti Ruismäki, Kalle Saarinen, Tommi Salonen, Martin Schuele, Jussi Sistonen, Mox Soini, Simo Säde, Eero Tamminen, Petra Tarkkala, Petri Tolppanen, Markku Ursin, Markku Vire, Luc Pionchon, and Jani Ylinen.

Additionally, thanks to all UI Designers in the UI team, and to all usability team members.

# 2 Hildon touch view

*Hildon 2.2 UI Style Guide* defines Views and Sub Views as the main UI for all Tasks (applications).

Views work in a tree-like hierarchy: there is a Root View that acts as a ‘root’ for the tree, while the subsequent Sub Views are branching down the hierarchy and always have a ‘back’ -button (‘<-‘), which takes the user one step back toward the main view, closing the Sub View. Root View has a ‘close’ button (‘X’) in the top right corner.

See Chapter 3, ‘UI modes’, for more information on how the UI modes affect the views.

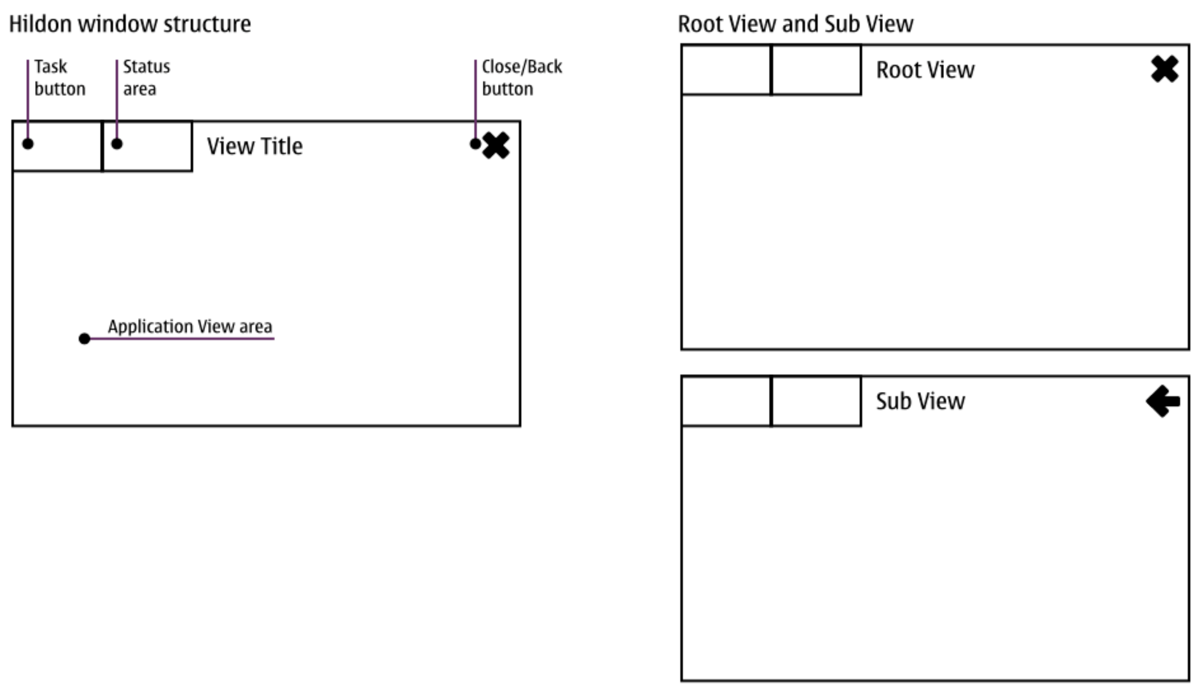


Figure 1: Hildon windows structure, Root View, and Sub View

## 2.1 Structure

The UI Framework at the top of the view consists of:

- *Tasks button*
- *Status area*
- *Title area* — Tapping on the Title area opens Hildon Touch view menu. The view title in Title area is truncated if it does not fit.
- One of two navigation buttons:

Figure 1: **Close** button, which is used in the Root View. It closes the task. The graphics for this button are defined in the Theme template. There is no logical string or icon string associated with it.

Figure 2: **Back** button, which is used in the Sub Views. It closes the task. The graphics for this button are defined in the Theme template. There is no logical string or icon string associated with it.

## 2.2 Default event actions

The following table shows the default actions for Hildon UI events.

| Area | Event | Action |
|---|---|---|
| Task Button | - | - |
| Status Area | - | - |
| Title Area | Button press | Shows the Hildon Touch view menu. |
| Title Area | Button release | |
| Close button (root view) | Button press | Shows a highlight effect for the button. |
| Close button (root view) | Button release | Closes the application. |
| Back button (sub view) | Button press | Displays a highlight effect for the button. |
| Back button (sub view) | Button release | Returns to previous view. |
| Back button (sub view) | Long press | Returns to root view. |

Table 1: Default event actions

## 2.3 Implementation

Touch view is implemented using the `HildonStackableWindow` class. Hildon window is not affected by this additional class.

### 2.3.1 The API for stacking HildonStackableWindows

New views can be created with `hildon_stackable_window_new()`. The stackable windows without a parent window are root views (there can be several) and, when visible, are shown as separate thumbnails in Task Switcher. When a stackable window has a parent window set, it becomes a part of the stack where the parent window is located. Once you add a new window to a stack, the parent one is hidden in the UI. Destroying a window shows the parent window again.

Each stack can be shown or hidden with `gtk_widget_show()` or `gtk_widget_hide()`.

Also, `hildon_program_go_to_root_window()` is provided to go back to the first window, thus closing all the other ones.

The title in the title area is set with `gtk_window_set_title()`.

# 3 UI modes

The Hildon 2.2 UI style defines some modes in the UI that change the way user interacts with a widget. The main purpose of these modes is to enable direct manipulation of items, while still allowing the user to select single or multiple items when necessary.

## 3.1 Normal mode

Normal mode is the standard mode that is used by default.

For views, the normal mode means that in lists and grids, tapping on an item is a direct action, causing a new sub view to open, or a command to be executed. The framework is visible in the normal mode.

For dialogues, the normal mode means that in lists and grids, tapping on an item is a direct action, causing a new sub dialogue to open, or a command to be executed.

The following table shows the List/Grid actions for events in normal mode.

| Area | Event | Action |
|---|---|---|
| Activatable item/area in List/Grid | Finger press | Highlight effect to show tapped item |
| Activatable item/area in List/Grid | Finger release | Activate tapped item |
| Activatable item/area in List/Grid | Dragging motion | Turn off the highlight effect |

Table 2: List/Grid actions for normal mode events

**Note:** For grids, the highlight should not span the whole tapped cell area, only the borders. See *Fremantle Master Layout Guide* for more information.

## 3.2 Do Not Disturb flag

The Do Not Disturb (DnD) is a variant of normal mode or full-screen mode, represented by flag that can be set by an application. This mode should be used ONLY when there is a very good reason for using it. Such reasons are, for example, playing a full screen video in Mediaplayer, taking a photo in Camera, or recording a video in Camera.

The flag should be only set when something is actively happening on the screen; for example, a video is playing full screen. When playing is stopped, the flag should be immediately removed.

When the DnD flag is set, it blocks *all* UI, except:

- Low battery warning
- Incoming call (dialogue)
- Power key menu
- Dialogues (system modals from other tasks, task modals from the current application)

The blocking of UI means that:

- Information banners are not shown at all.
- Auto lock is *not* allowed to activate.
- Screen dimming/blanking is *not* allowed to activate.

The DnD flag does not affect sounds, vibration, or LED lights.

### 3.2.1 Implementation

The DnD flag can be activated and de-activated with `hildon_gtk_window_set_do_not_disturb()`.

## 3.3 Edit mode

Edit mode is used only in views, not in dialogues. It has two main purposes:

- Providing multiple selection functionality in list or grid
- Providing standard UI for editing single content item; for example, an image

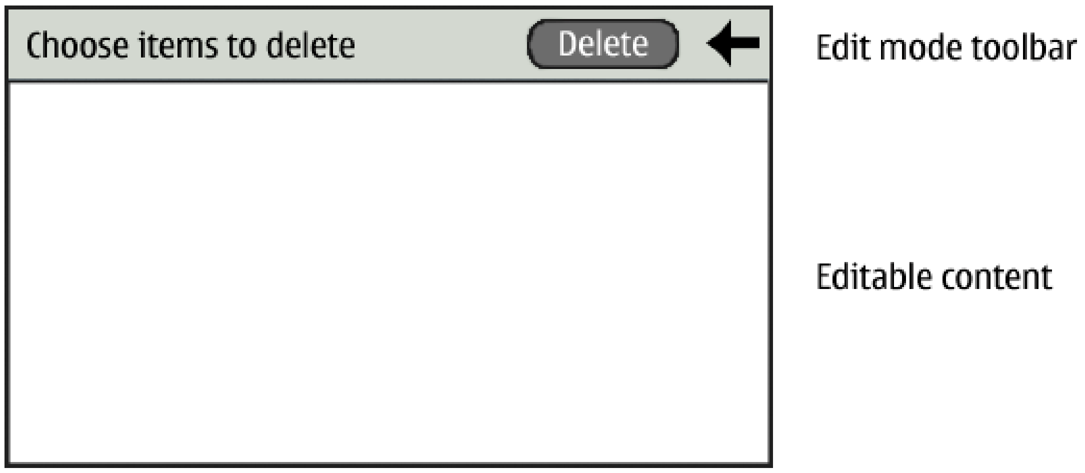


Figure 2: Edit mode for a view

In Edit mode, dialogues, banners, and such are shown as in normal views. If a dialogue or an incoming event results in switching to another task or Home view, Edit mode is shown in the Task Switcher for the related task. This is also the same as with normal views.

**Note:** It is possible to use Hildon Touch Progress indicator also in Edit mode, if the processing in Edit mode could take a long time.

### 3.3.1 Structure

- Edit Mode Toolbar widget (see Section 7.2, 'Hildon touch Edit Mode Toolbar')
- Editable content (for example, a Hildon Touch list in multiple selection mode)

Additionally, for the Multiple selection mode, when there are no selected items, the command button should still stay active, but simply do nothing when pressed. No banners or other similar items are shown.

### 3.3.2 Implementation

Edit mode is implemented with a sub view, using Hildon Touch View set in the Full screen mode (see Section 3.6, 'Full-screen mode'). In Edit mode, the normal framework is hidden, giving the task the whole screen area. A special Hildon Touch Edit Mode Toolbar widget is used at the top of the view, allowing the user to continue or return to the previous view.

## 3.4 Single selection mode

Single selection mode is typically used only in lists and grids in dialogues. In views, the Edit mode is usually used instead.

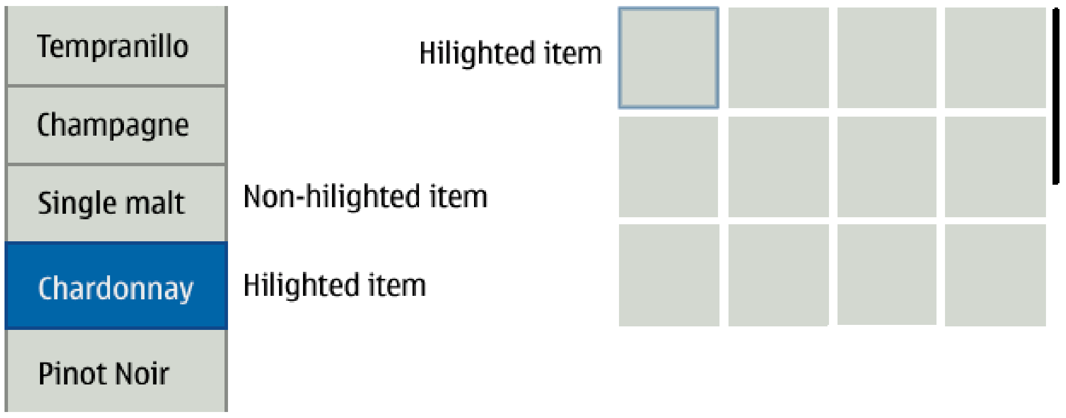


Figure 3: Single selection mode for a list

### 3.4.1 Structure

Each list item text is horizontally center-aligned by default. If a more complex content item than simple text is used (for example, icons + text), then content should be left-aligned by default.

### 3.4.2 Behaviour

In the Single selection mode, tapping on a list item does not activate a command, or open a subdialogue, but instead selects the tapped item:

- The tapped item is always selected, even when it has already been selected before.
- There is always one selected item. If the user has not tapped any item, and there is no pre-existing selection value, then the first item in the list or grid is selected.
- No more than one item in a single list or grid can be selected at any time.
- If an item other than the currently selected item is tapped, then the currently selected item is de-selected and the tapped item is selected.

**Note:** When a list is viewed in the Single selection mode, the list should automatically scroll to show the current selection, when the list is initially shown.

| Area | Event | Action |
|---|---|---|
| Activatable item/area in List/Grid | Finger press | Highlight effect to show tapped item |
| Activatable item/area in List/Grid | Finger release | Select the tapped item |
| Activatable item/area in List/Grid | Dragging motion | Remove the highlight effect (but do not change any selection) |

Table 3: List/Grid actions for single selection mode events

### 3.4.3 Implementation

The single selection mode is only available in the `GtkTreeView` and `GtkIconView` widgets, if the Hildon mode style property is set to `HILDON_FREMANTLE` (this is handled by the theme).

Lists must be created with `hildon_gtk_tree_view_new(HILDON_UI_MODE_EDIT)` or `hildon_gtk_tree_view_new_with_model(HILDON_UI_MODE_EDIT, ...)`. Also, set the selection mode with `gtk_tree_selection_set_mode(..., GTK_SELECTION_SINGLE)`.

Grids must be created with `hildon_gtk_icon_view_new(HILDON_UI_MODE_EDIT)` or `hildon_gtk_icon_view_new_with_model(HILDON_UI_MODE_EDIT, ...)`. Also, set the selection mode with `gtk_icon_view_set_selection_mode(..., GTK_SELECTION_SINGLE)`.

To read the value the user has selected, use the following code example:

```
_my_button_callback (......) {
...
/* You need to have access to the treeview here */
selection = gtk_tree_view_get_selection (treeview);
gtk_tree_selection_get_selected (selection, NULL, iter);
/* Now iter points to the selected item in the tree view model
(you can get it using gtk_tree_view_get_model */
/* do something with the selection */
....
}
```

Example 1: Single selection mode implementation

## 3.5 Multiple selection mode

Multiple selection mode is typically used by lists and grids in both views (Edit mode) and dialogues.

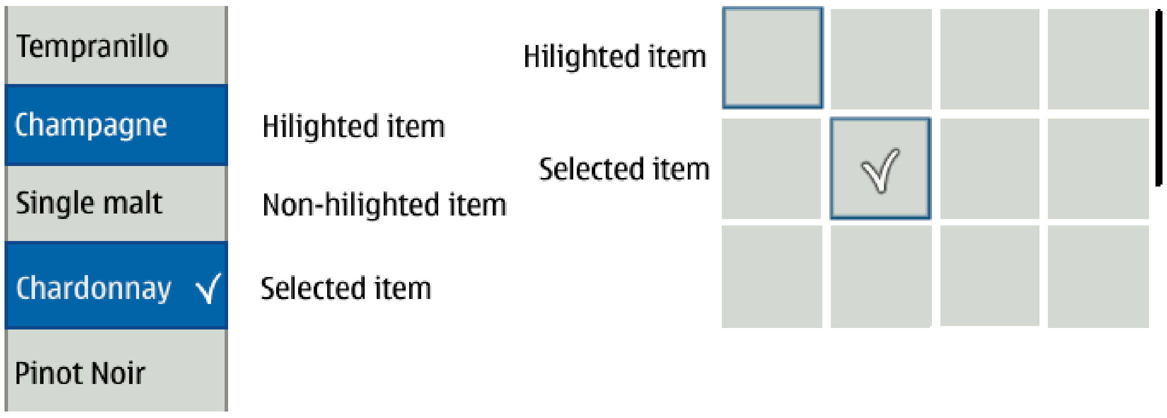


Figure 4: Multiple selection mode for a list

### 3.5.1 Structure

Each list item text is horizontally left-aligned by default. The tick-mark on all the selected items is right-aligned.

### 3.5.2 Behaviour

In the Multiple selection mode, tapping on a list item does not activate a command, or open a subdialogue, but instead selects the tapped item:

- When an item is tapped, it acts like a toggle: the item's state is changed from selected to deselected or vice versa.
- There are no items selected by default.
- There can be more than one item selected, in a single list or grid.
- User can deselect any item so that there are no items selected in the list.
- If any item other than the currently selected items is tapped, the other items selected keep their state.

Additionally, when there are no selected items, buttons that have actions on selected items should still stay active, but simply do nothing when pressed. No banners or other such items are shown.

| Area | Event | Action |
|---|---|---|
| Activatable item/area in List/Grid | Finger press | Highlight effect to show tapped item |
| Activatable item/area in List/Grid | Finger release | Select or deselect the tapped item, depending on the rules described above. |
| Activatable item/area in List/Grid | Dragging motion | Remove the highlight effect (but do not change any selections) |

Table 4: List/Grid actions for multiple selection mode events

**Note:** For Grids, the selection should be highlighted with special graphics on the selected grid cell background and border.

### 3.5.3 Implementation

The multiple selection mode is only available in the `GtkTreeView` and `GtkIconView` widgets, if the hildon-mode style property is set to `HILDON_FREMANTLE` (this is handled by the theme).

Lists must be created with `hildon_gtk_tree_view_new(HILDON_UI_MODE_EDIT)` or `hildon_gtk_tree_view_new_with_model(HILDON_UI_MODE_EDIT, ...)`. Also set the selection mode with `gtk_tree_selection_set_mode(..., GTK_SELECTION_MULTIPLE)`.

Grids must be created with `hildon_gtk_icon_view_new(HILDON_UI_MODE_EDIT)` or `hildon_gtk_icon_view_new_with_model(HILDON_UI_MODE_EDIT, ...)`. Also set the selection mode with `gtk_icon_view_set_selection_mode(..., GTK_SELECTION_MULTIPLE)`.

Application developers do not have to deal with drawing the tick marks, the `GtkTreeView` and `GtkIconView` widgets automatically take care of displaying the tick mark if they have been set in the correct mode.

To read the value user has selected, use the following code example:

```
_my_button_callback (......) {
...
/* You need to have access to the treeview here */
selection = gtk_tree_view_get_selection (treeview);
gtk_tree_selection_get_selected (selection, NULL, iter);
/* Now iter points to the selected item in the tree view model
```

```
(you can get it using gtk_tree_view_get_model */
/* do something with the selection */....
}
```

Example 2: Multiple selection mode implementation

## 3.6 Full-screen mode

Full-screen mode is changed from earlier Hildon versions. It is not a toggle like before, but rather a special case: an initial view when viewing single content item, such as an image or video. When the user taps the centre of the screen, the normal framework controls slide to view.

See *Hildon 2.2 UI Style Guide*, Section 9.4, for more information.

### 3.6.1 Special cases and legacy support

When it is not possible to make the UI Framework visible by tapping the content area, because the content area has too many or not predefined interactive elements (for example, Web browser and Games), a special full-screen button can be used for bringing the UI Framework back onscreen.

The two approaches for this functionality are:

- Using the full-screen toolbar button (for example, web browser) for full screen. Use the `general_fullsize` icon (full colour).
- Using full screen overlay (back-)button, with the button positioned on top right corner (for example, legacy games). Use the `general_overlay_back` icon (1bit colour).

# 4 Hildon Touch Pannable Area

Pannable Area is a container widget that can be ‘panned’ (scrolled) up and down using the touch screen with fingers. The widget has no scrollbars, but rather shows small scroll indicators to give an idea which part of the content is visible at the time. The scroll indicators appear when the list is initially shown and when a dragging motion is started on the pannable area. The indicators fade out automatically after one (1) second when the screen is not touched.

The scrolling is ‘kinetic’, meaning the motion can be ‘flicked’ and it continues from the initial motion by gradually slowing down to an eventual stop. The motion can also be stopped immediately by pressing the touch screen over the pannable area. When the user pans over the edge of the list, there is a bump effect for the list.

By default, the pannable area can be panned only in vertical direction.

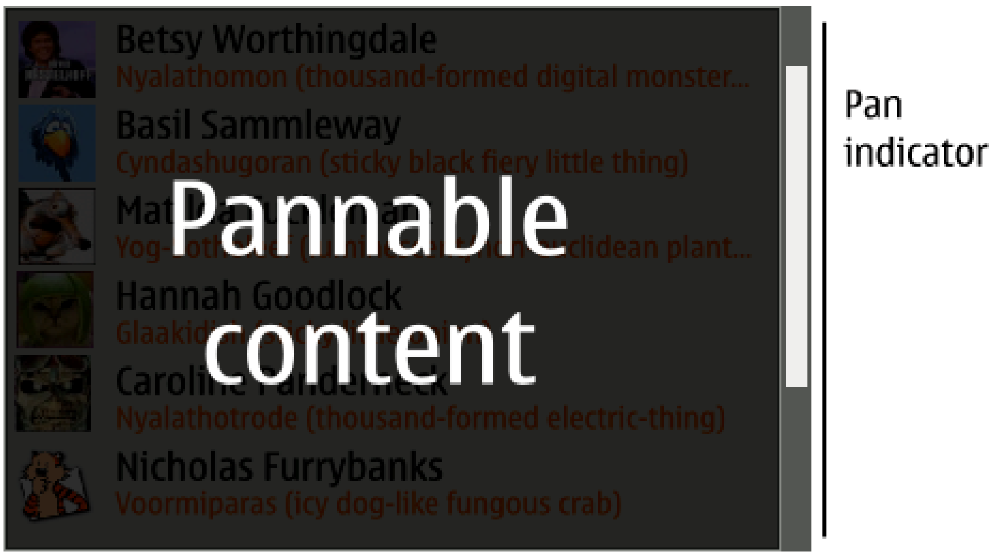

Betsy Worthingdale
Nyalathomon (thousand-formed digital monster...
Basil Sammleway
Cyndashugoran (sticky black fiery little thing)
Hannah Goodlock
Nyalathotrode (thousand-formed electric-thing)
Nicholas Furrybanks
Voormiparas (icy dog-like fungous crab)
Pannable content
Pan indicator

Figure 5: Pannable  Area widget with pan indicator

## 4.1 Behaviour

The following table describes pannable area event actions.

Table 5: Pannable area event actions

| Area | Event | Action |
|---|---|---|
| Content area | Touch press and dragging motion | Pan list towards the direction dragged, at the speed of the drag |
| ^ | ^ | Scroll indicator fades into view, shows position |
| Content area | Touch release after dragging motion | Continue present direction and speed in a decaying motion, slowly coming to a stop |

| Area | Event | Action |
|---|---|---|
| ^ | ^ | Scroll indicator fades out after a timeout once moving stops |
| Content area (in motion) | Button press without motion | Stop the scrolling immediately |
| ^ | ^ | Scroll indicator fades out after a timeout |
| Content area (no motion) | Button press | Pass through to container widget, should invoke a highlight effect on the selected item |
| Content area (no motion) | Button release | Pass through to container widget, should activate the selected item |
| Content area (while highlight on) | Dragging motion starts | Highlight should be "cancelled", panning is started |
| Scroll indicator | Button press | - |
| ^ | Button Release | - |
| ^ | Dragging motion | - (scroll indicator does not need to work like a scrollbar, it's just an indicator and too narrow to be usable anyway) |

The following figure presents an interaction diagram for the Pannable Area widget.

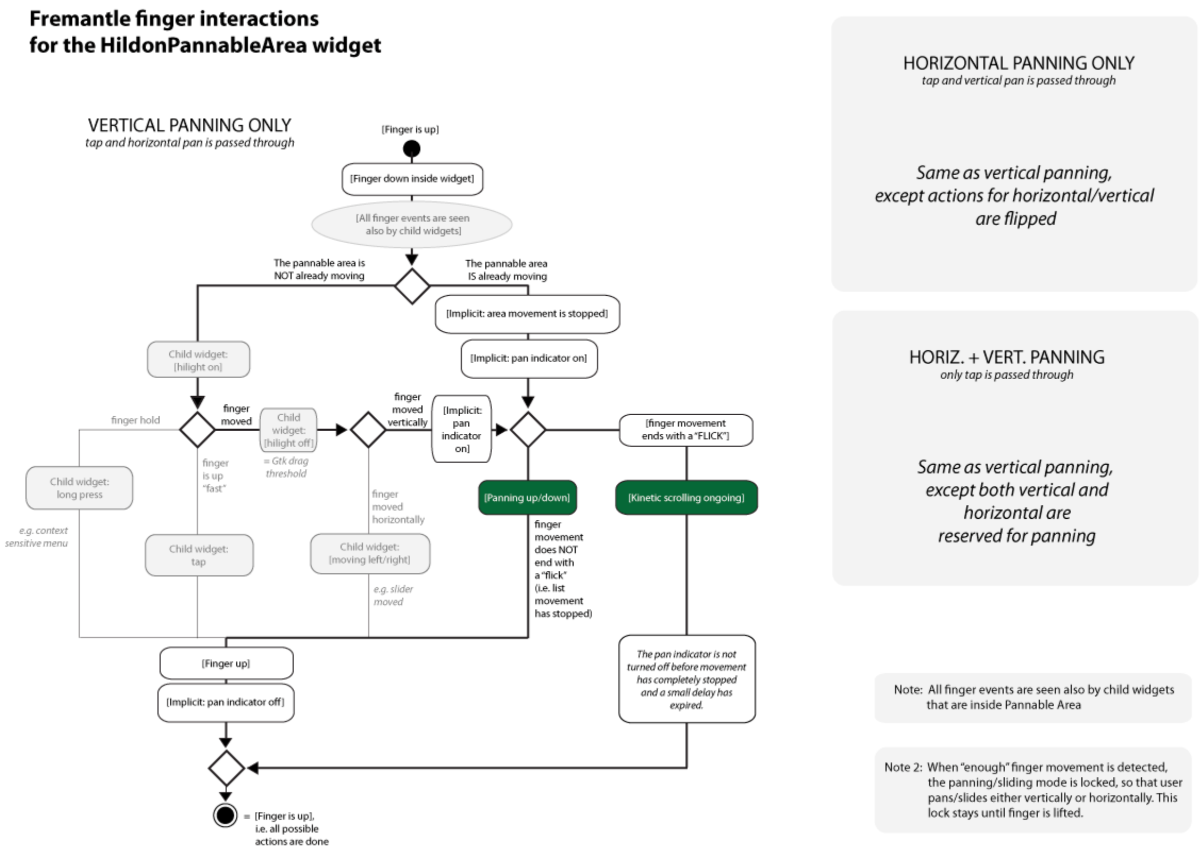

Fremantle finger interactions
for the HildonPannableArea widget
VERTICAL PANNING ONLY
tap and horizontal pan is passed through
[Finger is up]
[Finger down inside widget]
[All finger events are seen also by child widgets]
The pannable area is NOT already moving
The pannable area IS already moving
[Implicit: area movement is stopped]
[Implicit: pan indicator on]
Child widget: [hilight on]
finger hold
finger moved
Child widget: [hilight off]
= Gtk drag threshold
finger is up "fast"
finger moved vertically
[Implicit: pan indicator on]
[finger movement ends with a "FLICK"]
[Panning up/down]
[Kinetic scrolling ongoing]
Child widget: long press
e.g. context sensitive menu
Child widget: tap
finger moved horizontally
Child widget: [moving left/right]
e.g. slider moved
finger movement does NOT end with a "flick" (i.e. list movement has stopped)
The pan indicator is not turned off before movement has completely stopped and a small delay has expired.
[Finger up]
[Implicit: pan indicator off]
= [Finger is up], i.e. all possible actions are done
HORIZONTAL PANNING ONLY
tap and vertical pan is passed through
Same as vertical panning, except actions for horizontal/vertical are flipped
HORIZ. + VERT. PANNING
only tap is passed through
Same as vertical panning, except both vertical and horizontal are reserved for panning
Note: All finger events are seen also by child widgets that are inside Pannable Area
Note 2: When "enough" finger movement is detected, the panning/sliding mode is locked, so that user pans/slides either vertically or horizontally. This lock stays until finger is lifted.

Figure 6: Pannable Area widget interaction diagram

### 4.2 Widgets supported inside the pannable area

The following list covers all the widgets that are supported as child-widgets inside the Hildon 2.2 pannable area. When a finger is pressed down on the pannable area, the Pannable Area widget should activate the highlight of the topmost child widget under the finger.

| Widgets with highlight | Widgets without highlights |
|---|---|
| GtkTreeview<br>A single list row/item is highlighted. | GtkLabel |
| GtkIconView<br>A single grid item is highlighted. | GtkImage |
| GtkButton<br>Also, the Hildon 2.2 'picker' buttons. | GtkDrawingArea<br>Used when drawing, for example with cairo. |
| GtkCheckbox<br>with hildon-mode set as `HILDON_FREMANTLE` (with a button background) | GtkProgressBar |
| GtkToggleButton | GtkHSeparator |
| GtkEntry<br>Note that while the highlight (the same as focus graphics) should appear during finger down, the actual focus (with the caret) should only activate on finger release. | GtkVSeparator |
| GtkTextView<br>Note that while the highlight (the same as focus graphics) should appear during finger down, the actual focus (with the caret) should only activate on finger release. | - |
| GtkHScale<br>Can be used inside vertical Pannable Area. | - |
| GtkVScale<br>Can be used inside horizontal Pannable Area. | - |

Table 6: Pannable area widgets

In addition, the following containers can be used for layout:

- `GtkHBox`
- `GtkVBox`
- `GtkTable`
- `GtkSizeGroup`

However, these containers should pass through the highlight to the top-most child-widget under the finger.

### 4.3 Implementation

Pannable Area is implemented using the `HildonPannableArea` class, which behaves much like `GtkScrolledWindow`:

```
GtkWidget *child = ...
GtkWidget *panarea = hildon_pannable_area_new ();
hildon_pannable_area_add_with_viewport (HILDON_PANNABLE_AREA (panarea),
GTK_WIDGET (child));
```

# 5 Hildon Touch List

Touch List View is a widget to display user content in a list. The widget should always be combined with the Pannable Area widget, so it can be scrolled with fingers in a swiping motion.

The widget can optionally have text labels as group titles.

See Chapter 3, 'UI modes', for more information on the possible modes in the list widget.

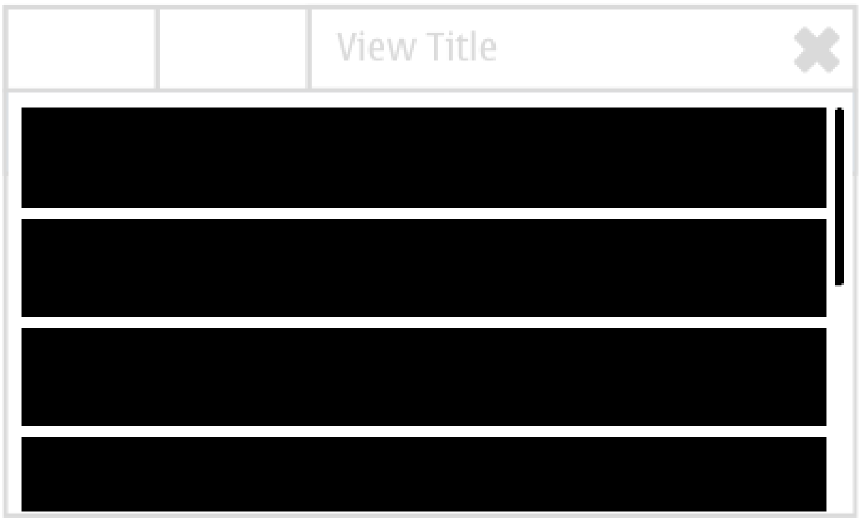


Figure 7: Touch List

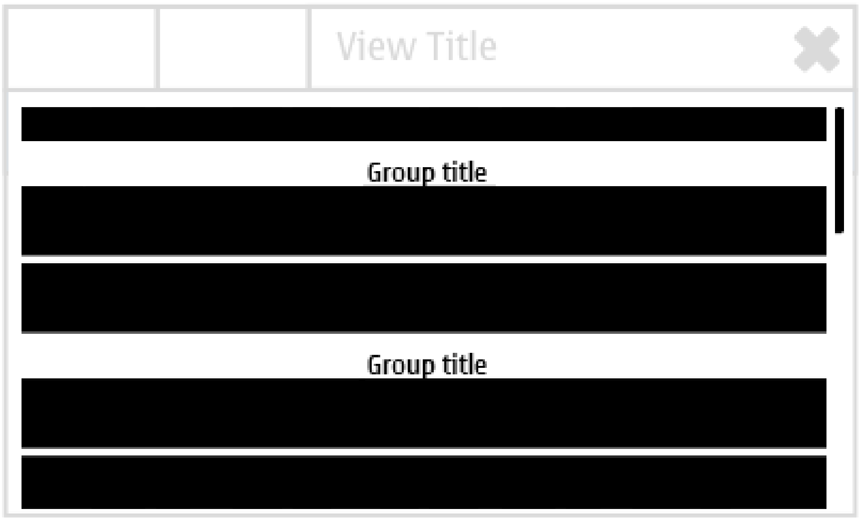


Figure 8: Touch List with group titles

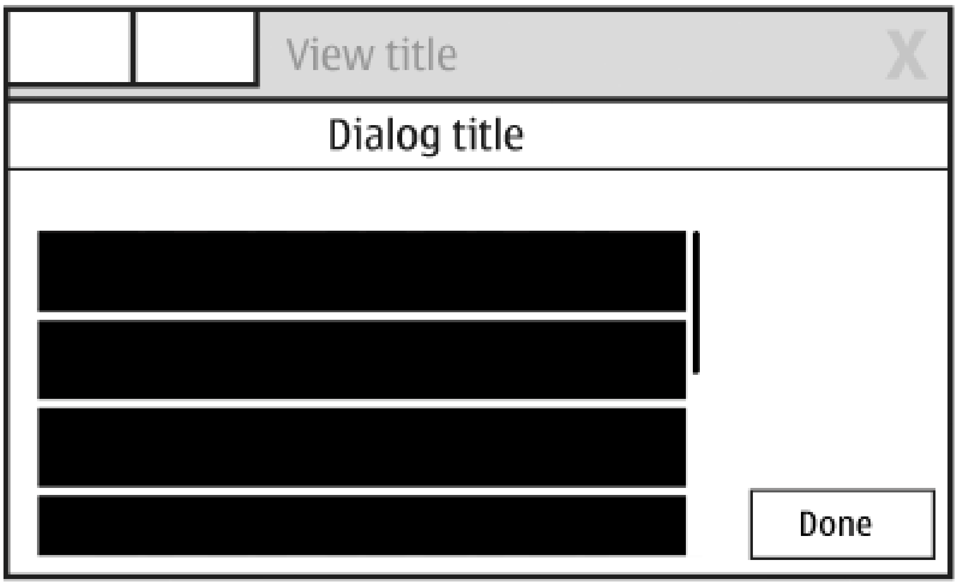


Figure 9: Touch List in the multiple selection mode and in a dialogue

## 5.1 Behaviour

The list has no 'focus' and the arrow keys cannot be used for scrolling or selecting anything in the list view. Instead, items in each row can be directly pressed with fingers and they are immediately activated.

The list consists of multiple rows that can have different kinds of content. The most common style is two columns, where on the left side there is an icon, a thumbnail, an avatar photo or some other information in a thumbnail-like representation, and the right column contains one or two lines of text. Items such as icons can have different actions than the text in the list.

There can be more than one active element on a single row. However, this behaviour needs to be explicitly provided in the application software.

*Hildon 2.2 UI Style Guide* emphasises direct manipulation, and that means there is no 'selection' visible by default. Tapping activates an action directly. This means that cursor keys are not supported for navigating the list. The list is intended to be used by fingers: pan directly with fingers to scroll the list, tap to activate a command or immediately open the item in a new view. See *Hildon 2.2 UI Style Guide* for more information about the user interface style and guidelines.

The list items can be grouped with group titles. The group title rows are smaller in height than the normal rows. The title text is normal text, with no background graphics. The text is centered and at the bottom of the row.

**Note:** When a list is inside Pannable Area, panning is only allowed *vertically*.

## 5.2 Implementation

Touch List is implemented using the `HildonPannableArea` class combined with `GtkTreeView`. The `TreeView` should be created using `hildon_gtk_tree_view_new(HILDON_UI_MODE_NORMAL)` or `hildon_gtk_tree_view_new_with_model(HILDON_UI_MODE_NORMAL, ...)`.

**Note:** A pannable treeview is only supported when the `HildonPannableArea` is the direct parent of `TreeView`. Containers such as `GtkHBox` widgets or sibling widgets such as buttons are not supported inside the same pannable area.

By default, each list row is one item as far as click events are concerned. If more than one active element is needed for a single row, the application needs to:

- Capture the click event and determine which row was clicked
- Calculate which element in the row was clicked
- Produce the action or command for that particular element

For icon view, the same basically applies, except that the two identifiers have different names: `HildonIconViewRowHeaderFunc` and `hildon_icon_view_set_row_header_func()`, respectively.

For custom cell renderers using various font sizes and colours, see Section 13.2, 'Aligning and theming complex button text'.

**Note:** For performance reasons, it is recommended that for complex list items (those with more elements than just text) a custom cairo cell renderer is used.

### 5.2.1 Row headers implementation

In `GtkTreeView`, row (or group) headers work in the same manner as row separators. A `RowHeaderFunc` has to be installed that is called to determine whether or not a given row is a row header. The `RowHeaderFunc` that must be implemented has the following definition:

```
typedef gboolean (*HildonTreeViewRowHeaderFunc) (GtkTreeModel *model,
GtkTreeIter *iter, gchar **header_text, gpointer data);
```

A TRUE return value indicates that `iter` is a row header. FALSE is returned if this is not the case. When TRUE is returned, `header_text` is expected to point to a string that should be the title of the row header. The `RowHeaderFunc` should be installed using `hildon_tree_view_set_row_header_func()`.

# 6 Hildon Touch Grid

Hildon Touch Grid is a widget to display the user content in a grid layout. The widget should always be combined with the pannable area widget for consistent user experience.

The grid has no 'focus' or keyboard interactions, instead each grid 'cell' can be directly pressed and it is immediately activated. The cell can contain either an image, an image with a label underneath, or custom content created by the application itself.

See Chapter 3, 'UI modes', for more information on the possible modes in the grid widget.

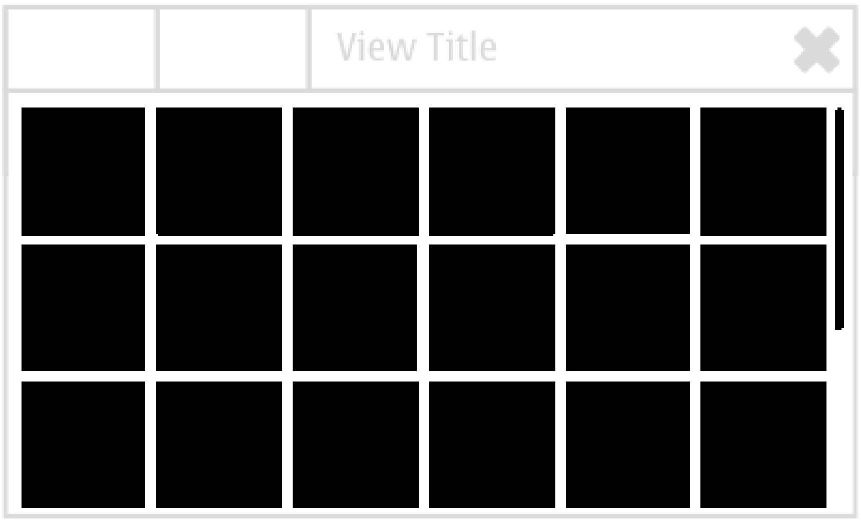


Figure 10: Grid displaying pannable content

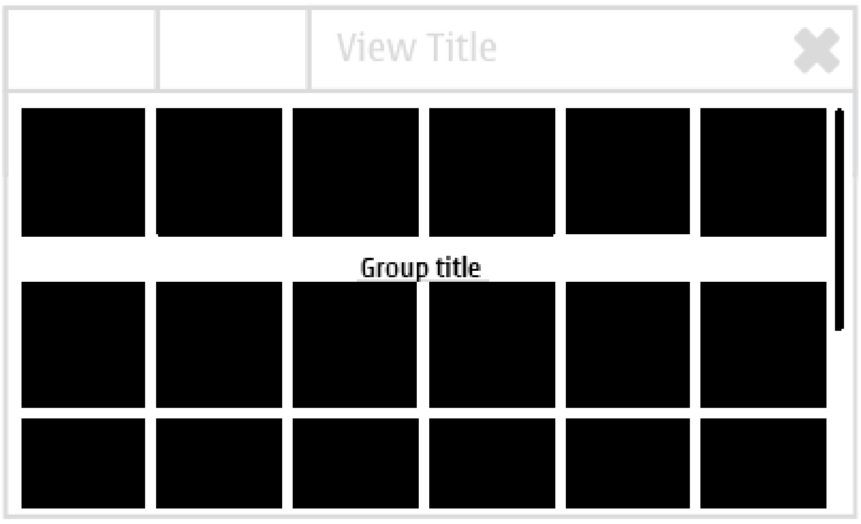


Figure 11: Grid with group titles

## 6.1 Behaviour

The grid is a 'flowing' layout of equal-sized cells that are arranged in as many columns as fits on the screen. The size of one content cell determines the number of columns that fit on the screen at once. The number of rows is determined by the number of items displayed. If the content does not fit on the screen at once, the widget allows the user to pan the content by fingers. Therefore the grid view always needs to be combined with a pannable area widget for kinetic scrolling.

Each grid item acts as an active area that immediately invokes an action on the tapped item (normal mode).

The grid view has no 'focus'. Arrow keys cannot be used for scrolling or selecting anything in the view.

The grid items can be grouped with group titles. The group title rows are smaller in height than the normal rows. The title text is normal text, with no background graphics. The text is centred and at the bottom of the row.

**Note:** Empty grid items and unloaded or partly loaded grid items should be shown visually empty, that is, totally transparent. For indicating a loading progress, use Hildon Touch Progress indication in the view title or in the dialogue title.

See *Hildon UI 2.2 Style Guide* for more information about the user interface style and guidelines.

**Note:** This needs a lot of prototyping and tweaking to get the timing and sensitivities right.

## 6.2 Implementation

Touch Grid is implemented using the `HildonPannableArea` class combined with `GtkIconView`. The `IconView` should be created using `hildon_gtk_icon_view_new(HILDON_UI_MODE_NORMAL)` or `hildon_gtk_icon_view_new_with_model(HILDON_UI_MODE_NORMAL, ...)`.

**Note:** Pannable icon view is only supported when the `HildonPannableArea` is the direct parent of `IconView`. Containers like `GtkHBox` widgets or siblings like buttons are not supported inside the same pannable area.

**Note:** For performance reasons, it is recommended that for complex list items (those with more elements than just text) a custom cairo cell renderer is used.

### 6.2.1 Row Headers implementation

In `GtkIconView`, row (or group) headers work on the same manner as row separators. A `RowHeaderFunc` has to be installed that is called to determine whether or not a given row is a row header. The `HildonIconViewRowHeaderFunc` that must be implemented has the following definition:

```
typedef gboolean (*HildonIconViewRowHeaderFunc) (GtkTreeModel *model,
GtkTreeIter *iter, gchar **header_text, gpointer data);
```

A TRUE return value indicates that `iter` is a row header. FALSE is returned if this is not the case. When TRUE is returned, `header_text` is expected to point at a string that should be the title of the row header. The `HildonIconViewRowHeaderFunc` should be installed using `hildon_icon_view_set_row_header_func()`.

# 7 Toolbars

This chapter describes the use of toolbars and similar UI elements in the UI.

## 7.1 Hildon Touch Toolbar

The toolbar is a basic opaque container that floats over the content area. By default, the toolbar is at the bottom edge of the screen.

Use of toolbars should be avoided. However, toolbars can be used in some special cases when there is only one content item visible (for example, when editing a single image).

Basic toolbars are not used for list and grid views. Instead, for some specific cases, embedded Hildon touch finger buttons can be used at the top of and inside the pannable area for lists and grids. See *Hildon 2.2 UI Style Guide* for more information.

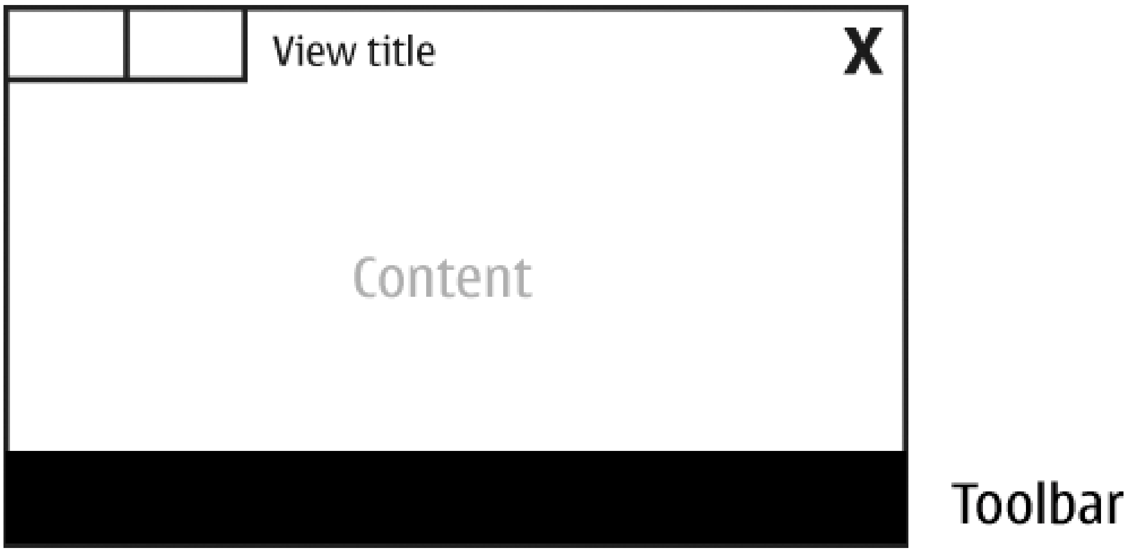


Figure 12: Toolbar

**Note:** There should be no menu commands or settings for hiding or showing toolbar. The toolbar is always shown.

Because toolbar is a simple container, the user cannot interact with it.

### 7.1.1 Implementation

Toolbar is implemented using the `GtkToolbar` class. To add a toolbar to window, use `hildon_window_add_toolbar()`. Toolbars in legacy applications also get the Fremantle style height. The behaviour is the same as in Hildon 2.1.

## 7.2 Hildon touch Edit Mode Toolbar

The Edit Mode Toolbar widget is the main control and navigation interface for the Edit mode.

The edit mode toolbar is a Hildon touch toolbar with specific contents. It is only used when UI framework is not visible, and is positioned at the top edge of the screen.

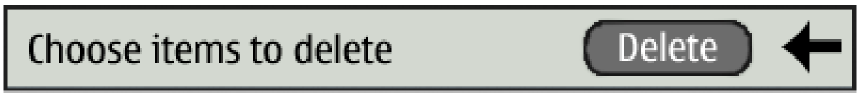


Figure 13: Edit Mode Toolbar

### 7.2.1 Structure

- *Description* — Application provided command instruction. For example, 'Choose images to delete'. The text is aligned to the left edge in the toolbar.
- *Command* — A button using the application-provided button title, or `wdgt_bd_done` by default. The button is aligned near and left of back navigation.
- *Back navigation* — A button behaving like back navigation in framework. Returns to the previous view, discarding any changes. The button is aligned to the right edge of the toolbar.

### 7.2.2 Implementation

Edit mode toolbar is implemented using the `HildonEditToolbar` widget. To add the Edit mode toolbar to a window, use `hildon_window_set_edit_toolbar()`.

# 8 Hildon Touch View Menu

View menu opens when the title area of a view is pressed. The menu opens from the very top of the screen, and always occupies the full width of the screen, sans a small border on both sides. The height is determined by the number of menu items present.

The area ‘outside’ the menu should be dimmed and blurred to visually separate the menu from the application underneath it. Pressing the dimmed area outside the menu closes the menu without invoking any action.

Pressing a menu item closes the view menu and performs the action associated with the item.

The menu can optionally include *one* set of grouped ‘view filter buttons’ to manipulate the application view's data representation; for example, to change the way a list of contacts is sorted, or whether to display items as grid or list.

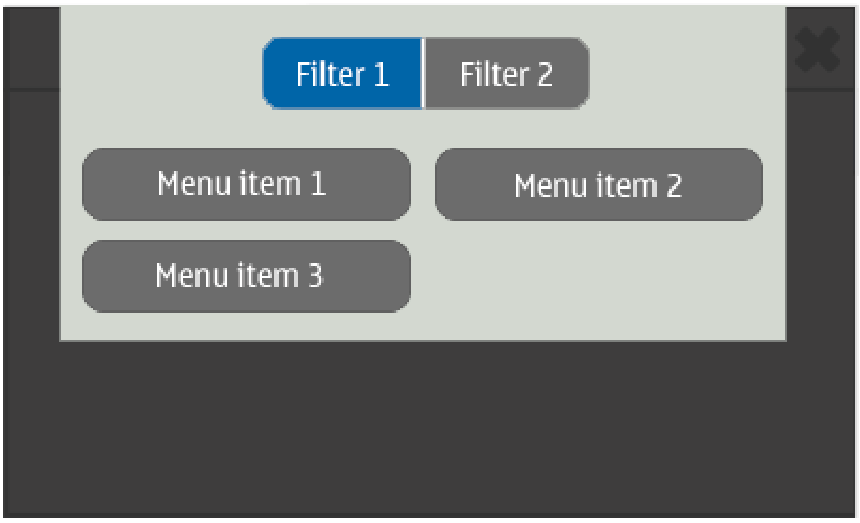


Figure 14: View menu

**Note:** The view menu does NOT pan. The amount of menu items is hard-limited to ten (= 5 items in two columns, when no filters). There are NO submenus.

### 8.1.1 Structure

- Optional filters as toggle buttons in a visually joined group (see Section 13.4, ‘Hildon Touch Toggle button’ for more information). Behave like radio buttons.

### 8.1.2 Behaviour

Each menu item represents an action that is invoked when pressed; essentially, the menu items are push buttons. The menu closes after a menu item is invoked.

Table 7: Touch View Menu actions

| Area | Event | Action |
|---|---|---|
| View Title in Title area | Button press | View Menu opens. |
| View Title in Title area | Button release | - |
| Area outside View menu | Button press | - |
| Area outside View menu | Button release | View menu closes. |
| Menu item | Button press | Menu item is highlighted. |
| Menu item | Button release | View Menu closes and the command is invoked. |
| Menu item | Dragging motion | Menu item is de-highlighted. Command is not activated. |
| Menu filter item | Button press | Menu item is highlighted. |
| Menu filter item | Button release | The filter is applied, menu is closed. |
| Menu filter item | Dragging motion | Menu item is de-highlighted, filter is not applied, menu is not closed. |

Additionally:

- Menu items can be hidden, but the items are never dimmed. This causes the layout to be automatically updated (that is, there are no empty slots in between the menu items).
- The menu should work in portrait mode using a single column.
- If the menu contains no items, then the menu should not open at all.

### 8.1.3 Implementation

Touch View Menu is implemented using the `HildonAppMenu` class. It can be attached to a window (see the example below) and it can also be invoked just by showing the widget:

```
gtk_widget_show (menu)
```

# 9 Hildon Touch Context Menu

Context menu is usually invoked via a long press over an item on the screen, such as by holding a finger over an image thumbnail. The menu should contain commands directly related to the chosen item.

The use of Context menus should be avoided. While offering advanced functionality for power users, it is considered as a hidden and inconvenient way of interacting with the UI. Regular Hildon Touch View Menus should be used instead whenever possible.

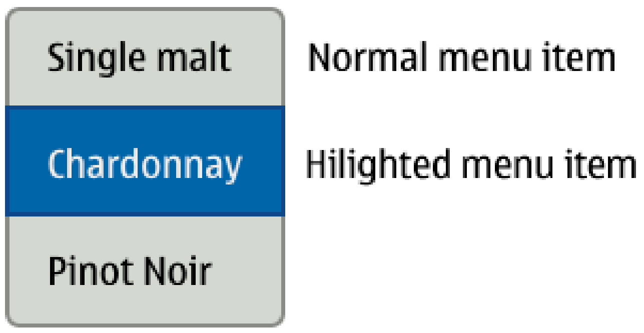


Figure 15: Context menu

## 9.1 Behaviour

In Hildon 2.2, the menu items are enlarged and the menu is not pannable. The number of menu items is limited to what fits on the screen at once, so this requires a careful selection of menu items. Submenus are not allowed. If the Context menu contains no items, then the menu should not open at all.

**Note:** While the functionality for submenus and scrolling by arrow buttons exists, submenus, and scrolling are not supported by the UI Framework.

There is no activation animation for the Context menu.

Table 8: Touch Context Menu actions

| Area | Event | Action |
|---|---|---|
| UI Element that has associated context menu | Finger press | UI element is highlighted. |
| UI Element that has associated context menu | Long press | Show Context menu. |
| UI Element that has associated context menu | Finger release after long press | Context menu stays on screen. |
| Area outside Context menu | Finger press | - |
| Area outside View menu | Finger Release | Context menu closes. |
| Menu item | Finger press | Menu item is highlighted. |

| Area | Event | Action |
|---|---|---|
| Menu item | Button release | Context Menu closes and the command is invoked. |
| Menu item | Dragging motion | Menu item is de-highlighted. Command is not activated. |
| Menu group item | Finger press | Group item is highlighted. |
| Menu group item | Finger release | Appropriate sub Context Menu is shown. |
| Menu group item | Dragging motion | Group item is de-highlighted. Command is not activated. |

### 9.2 Implementation

Touch Context Menu is implemented using the `GtkMenu` class, created with `hildon_gtk_menu_new()`. Hildon Touch Context Menu has the same behaviour as the GTK Contextual Menu widget in Hildon 2.1.

# 10 Banners and notes

Banners and notes are changed from Hildon 2.1 in several ways. Since the full width of the screen is available for applications, the banners and notes use the whole screen horizontally. Also, the buttons are moved to the right side of the note, giving more vertical space for the content. The right edge of the note is reserved for the buttons, which should be vertically aligned to the bottom of the note, and should expand horizontally so that all buttons are of equal width.

Also, the **Cancel** button is removed from the notes, as specified in *Hildon 2.2 UI Style Guide*. The part of the screen that is outside of the note is dimmed and blurred. Clicking on this dimmed area outside the note closes it.

**Modality of notes**

Notes can be either system or task modal. See Chapter 11, 'Dialogues' for more information.

Table 9: Functionality of notes

| Area | Event | Action |
|---|---|---|
| Area outside the note (in the blurred/dimmed background) | - | - |
| Area outside the note (in the blurred/dimmed background) | Finger release | Closes the note. |

## 10.1 Hildon Touch Information banner

Information Banners show and hide by themselves, there is no user interaction and they work exactly as before. The style is different, however; the current understanding is to have them as horizontal stripes under the view title area.

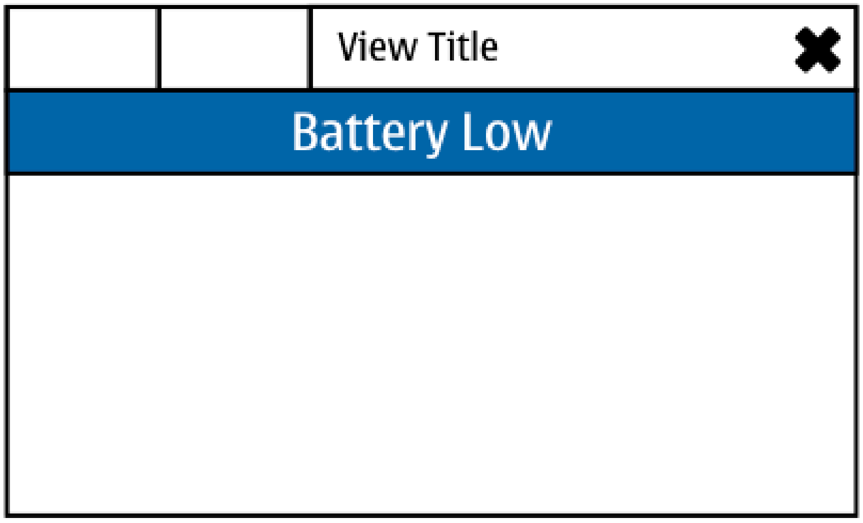


Figure 16: Information banner

No icons should be used with the information banner.

#### 10.1.1 Implementation

The information banner is implemented with the `HildonBanner` class. See http://maemo.org/api_refs/5.0/hildon/HildonBanner.html. This is the same function as used in Hildon 2.1.

### 10.2 Hildon Touch Information note

Compared to earlier specifications, the **OK** button is removed from the information note. The note is closed by tapping anywhere in the note area. As with the standard way, it is also possible to close the note by tapping the area outside of the note. The dialogue fills the whole width of the screen like a banner across the application view.

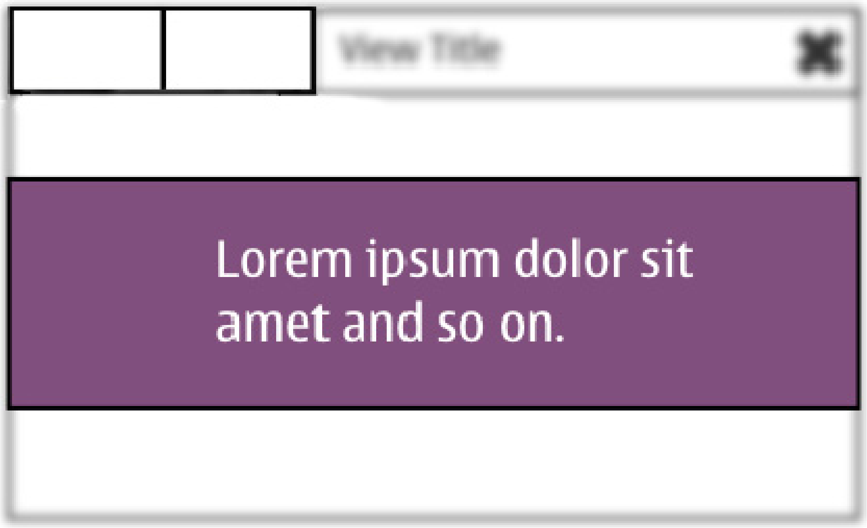


Figure 17: Information note

No icons should be used with the information note.

#### 10.2.1 Implementation

The information note is implemented with `hildon_note_new_information()`. See http://maemo.org/api_refs/5.0/hildon/HildonNote.html. This is the same function as used in Hildon2.1.

### 10.3 Hildon Touch Confirmation note

The confirmation note does not follow the ‘cancel a dialogue by tapping outside it’ design. Since a confirmation note is always used to represent an important choice to the user, both alternatives need to be shown clearly.

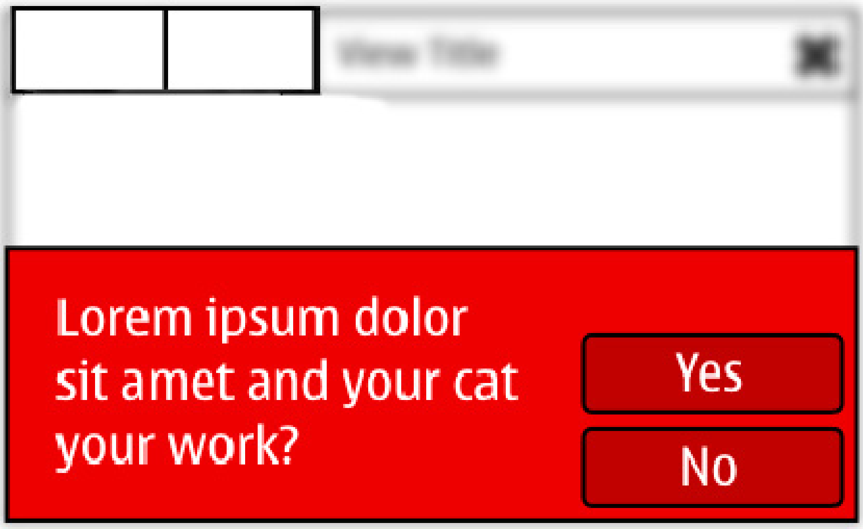


Figure 18: Confirmation note

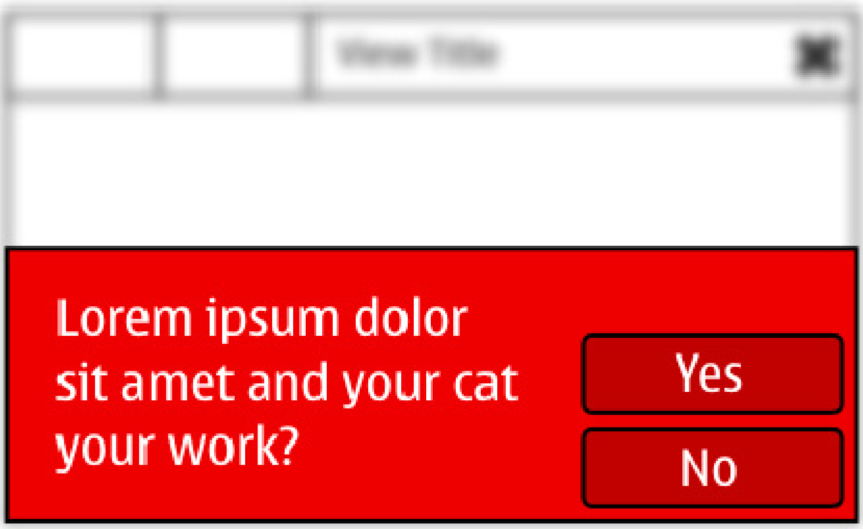


Figure 19: Confirmation note with system modal dialogue

Confirmation Note needs a visual hint of importance in its theming. This needs to be designed in the specification phase.

The buttons in the confirmation note are:

- *wdgt_bd_yes* — Accepts, closes the note and continues the process.
- *wdgt_bd_no* — Declines, closes the note and interrupts the process.

No icons or images should be used with the confirmation note.

Where a custom confirmation note is needed (if there are different buttons, for example), it can be specified separately.

### 10.3.1 Implementation

The confirmation note is implemented with `hildon_note_new_confirmation()`. See http://maemo.org/api_refs/5.0/hildon/HildonNote.html. This is the same function as used in Hildon 2.1.

## 10.4 Hildon Touch Incoming event

Incoming events appear for new communication events, such as chat messages, emails, or SMS. They slide in from the top of the screen, and stay on the screen for a short while, and then go away. They accept click events while they are on the screen, enabling the user to open the event in question.

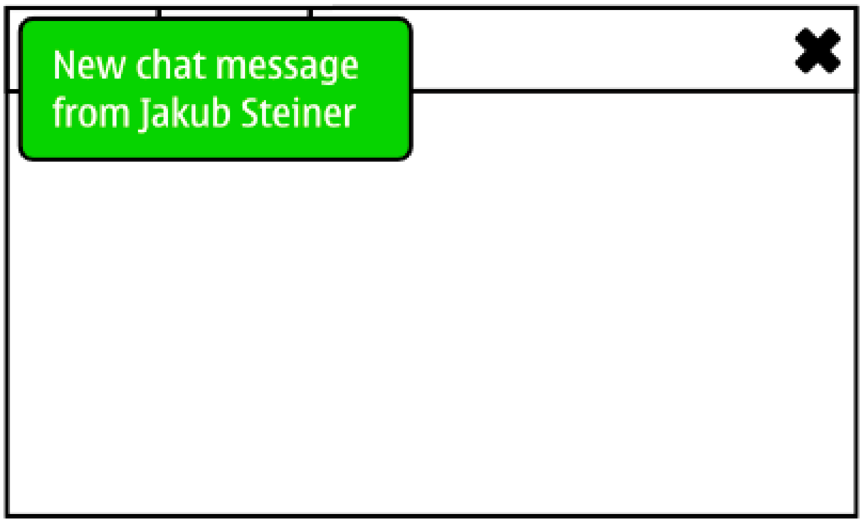


Figure 20: Incoming event

### 10.4.1 Implementation

The incoming event is implemented in the UI Framework in the code responsible for the notification service. A D-Bus API is provided for showing these kinds of notifications.

# 11 Dialogues

Dialogues are changed from Hildon 2.1 in several ways. Since the full width of the screen is available for Tasks, the dialogues use the whole screen horizontally. Also, the buttons are moved to the right side of the dialogue, giving more vertical space for the content.

Both legacy (Hildon 2.1) and new style (Hildon 2.2) applications are using the new style dialogues.

## 11.1 Modality of dialogues

If the dialogue is task modal, the Platform UI (Task button and Status area) are visible on top and can be used normally to switch between tasks. The note just covers the task view it is part of, but does not prevent switching to another task while it is open.

If the dialogue is system modal, both the Task button and Status area are blurred and dimmed. They are not active while the dialogue is open and task switching is not possible until the system modal dialogue is first deal with.

## 11.2 Hildon Touch Dialogue

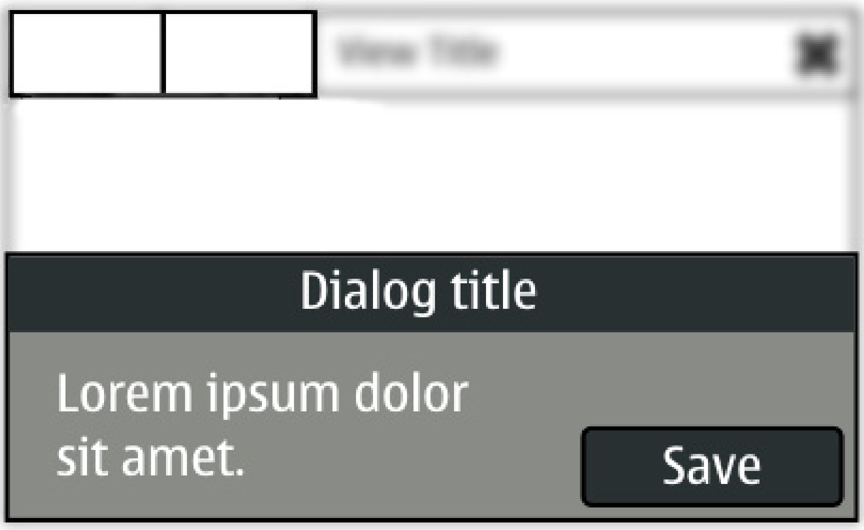


Figure 21: Touch Dialogue

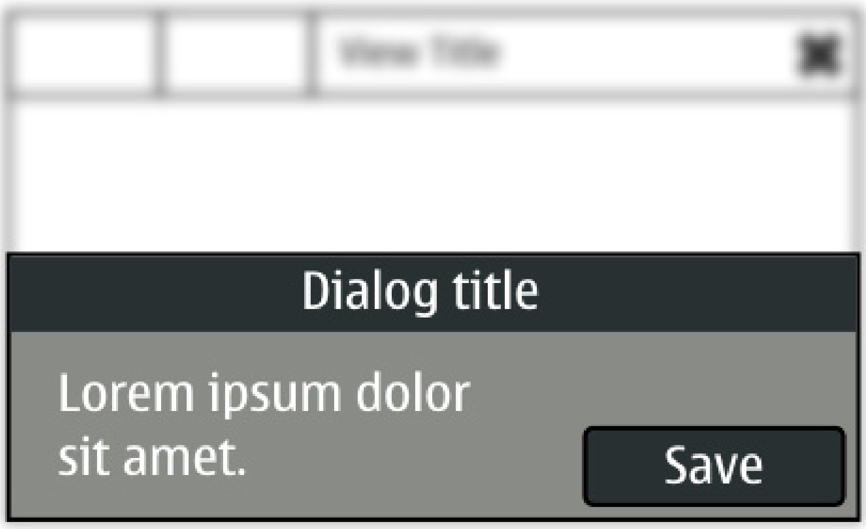


Figure 22: Touch Dialogue with system modal dialogue

### 11.2.1 Structure

Hildon Touch Dialogue is positioned on the bottom of the screen, and it uses the whole width of the screen horizontally. Vertically it occupies as much space as is needed by its content. The maximum size is the full height of the screen, minus the application title area and some empty space to allow cancellation of the dialogue. See *Fremantle Master Layout Guide* for exact dimensions and details.

The touch dialogue buttons are placed on the right side, and they align to the bottom of the dialogue, and expand horizontally to fill the button area so that all buttons are of equal width.

Also, there is no **Cancel** button in the dialogues, as specified in *Hildon 2.2 UI Style Guide*. The part of the screen that is outside of the dialogue is dimmed and blurred. Clicking on this dimmed area outside the dialogue closes the dialogue.

| Area | Event | Action |
|---|---|---|
| Area outside the dialogue (in blurred/dimmed background area) | Finger press | - |
| Area outside the dialogue (in blurred/dimmed background area) | Finger Release | Closes the dialogue |
| Area inside the dialogue | Finger press | - (inside the dialogue, for any child widget under the finger that supports highlight, show highlight) |
| Area inside the dialogue | Finger Release | - (inside the dialogue, for any child widget under the finger, do appropriate action) |

Table 10: Touch Dialogue actions

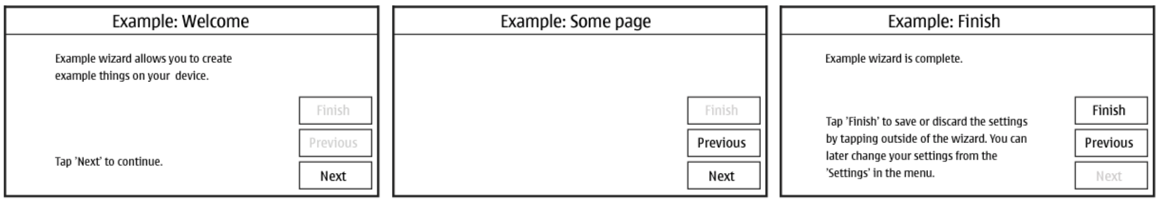
### 11.2.2 Implementation

The dialogues are implemented with the `GtkDialog` class.

**Note:** Dialogue buttons should be added with one of the following commands: `gtk_dialog_new_with_buttons()`, `gtk_dialog_add_button()` or `gtk_dialog_add_buttons()`.

## 11.3 Hildon Touch Wizard

Same as Hildon Touch Dialogue, but has predefined content.


Example: Welcome

Example wizard allows you to create example things on your device.

Tap 'Next' to continue.

Example: Some page

Example: Finish

Example wizard is complete.

Tap 'Finish' to save or discard the settings by tapping outside of the wizard. You can later change your settings from the 'Settings' in the menu.

Figure 23: Touch wizard example

### 11.3.1 Structure

Same as normal dialogue, the wizard is positioned at the bottom of the screen, there are three standard buttons `wdgt_bd_finish`, `wdgt_bd_previous` and `wdgt_bd_next` (from top to down).

- First page of the wizard has the title ‘Welcome’ and introductory text.
- Last page of the wizard has the title ‘Finish’ and tells user that the wizard has been completed and how to modify the settings later.
- Input fields: Whenever the input data has character limitations, they must be explicitly defined. When the user attempts to enter data exceeding the defined limitation, show Max. number of characters reached [WID-INF036].
- **Finish** button:

Figure 3: The button stays dimmed (cannot be pressed) until all required data has been entered by the user into the pages of the wizard dialogue.

Figure 4: When button is enabled and pressed, all user-entered data must be validated and checked.

Figure 5: If the button has been enabled and the user removes data from a required field, the button is dimmed.

Figure 6: If there are no errors, close the wizard.

- **Previous** button:

Figure 7: The button is dimmed only on the first page of the wizard.

Figure 8: When the button is pressed, no validation is made for the current page. Any data entered into the current page is retained.

- **Next** button:

Figure 9: The button is dimmed on the last page of the wizard.

Figure 10: If a page contains mandatory values that are empty, the button is dimmed until the mandatory values are filled.

Figure 11: When button is pressed, all user-entered data in the current page must be validated and checked. If data is valid, then the next page is shown.

- The wizard is cancelled by tapping outside of the wizard dialogue.

All the wizard pages use the `<wizard name> + ": " + <page title>` format. For example, `"Bluetooth: User details"`.

| Description | Logical name | UI string |
|---|---|---|
| Default title text for wizard’s welcome screen, where %s is the name of the wizard | ecdg_ti_wizard_welcome | "%s: Welcome" |
| Default descriptive text string for wizard's welcome screen, where the first %s is the name of the wizard and second the summary | ecdg_fi_wizard_description | "%s wizard allows you to %s.\n\nTap 'Next' to continue." |
| Wizard Button label | wdgt_bd_finish | "Finish" |
| Wizard Button label | wdgt_bd_previous | "Previous" |
| Wizard Button label | wdgt_bd_next | "Next" |

Table 11: Touch Wizard UI strings

### 11.3.2 Implementation

The wizard is implemented with the `HildonWizardDialog` class. See http://maemo.org/api_refs/5.0/hildon/HildonWizardDialog.html. This is the same function as used in Hildon 2.1.

# 12 Text and images

In Hildon 2.2, the standard widgets for showing basic text and images are the same as in Hildon 2.1.

*Hildon 2.2 UI Style Guide* introduces a couple of new usage scenarios: using text with shadow and visually indicating a tappable image with shadow.

## 12.1 Basic text

The basic text is used the same way as in Hildon 2.1. By default the text does not have a shadow.

The text can optionally contain shadow, but it needs to be explicitly defined.

### 12.1.1 Implementation

Implemented using the `GtkLabel` class. Note that text strings are always truncated when needed, ellipsising is not used.

For font sizes and font colours, use `hildon_helper_set_*`:

```
hildon_helper_set_logical_font(widget, "SmallSystemFont");
hildon_helper_set_logical_color(widget, GTK_RC_FG, GTK_STATE_NORMAL,
"SecondaryTextColor");
```

For text shadow, use cairo to create a standard pango layout with the appropriate attributes and show the text as follows:

```
cairo_set_source_rgba (cr, 0.2, 0.2, 0.2, 0.8); // Dark grey 80% opacity
cairo_move_to(cr, x + 1, y + 1);
pango_cairo_show_layout (cr, pango_layout);
cairo_stroke (cr);
cairo_set_source_rgb (cr, 1,1,1); // White
cairo_move_to(cr, x, y);
pango_cairo_show_layout (cr, pango_layout);
cairo_stroke (cr);
```

## 12.2 Basic image

The basic image is used in the same way as in Hildon 2.1. By default the image is not tappable and does not have shadow.

The image can optionally be tappable or activatable, so that it opens a new view or dialogue. When an image is tappable, it *must* have a shadow. This needs to be explicitly defined.

### 12.2.1 Implementation

Implemented using the `GtkImage` class.

For a tappable image shadow, use the `GtkDrawingArea` widget in place of `GtkImage`. From the drawing area, do the following:

```
cairo_t *cr;

cairo_surface_t *cr_surface;
cr = gdk_cairo_create(widget->window);
cairo_rectangle(cr, x+2, y+2, w, h); // Image height and width and
offset x/y should be specified
```

```
cairo_set_source_rgba(cr, 0,0,0,.5); // 50% black shadow
cairo_fill(cr);
cr_surface = cairo_image_surface_create_from_png(slice->icon); // It may
be necessary to scale the image using cairo transforms [1]
cairo_translate(cr, x, y);
cairo_rectangle(cr, 0, 0, w, h); // Create a clipping rectangle at the
translated co-ordinates
cairo_clip(cr);
cairo_set_source_surface(cr, cr_surface);
cairo_paint();
```

Example 3: Tappable image shadow implementation

See http://www.cairographics.org/manual/cairo-Transformations.html for more information.

# 13 Buttons

This chapter lists all widgets that are based on the style and interaction of a button. This includes regular buttons, toggle buttons, and a check box button.

## 13.1 Hildon Touch Finger button, Thumb button

Hildon Touch Buttons are button widgets designed for finger-use with touchscreen input. There are two button sizes defined for UI design: Hildon Touch Finger button and Hildon Touch Thumb button. Finger Button is the default size to be used, Thumb Button is useful if a very large control is needed. The legacy toolkit additionally has a stylus-sized button but its use is discouraged as it is not usable with fingers.

See *Fremantle Master Layout Guide*, Chapter 6, for the exact measurements.

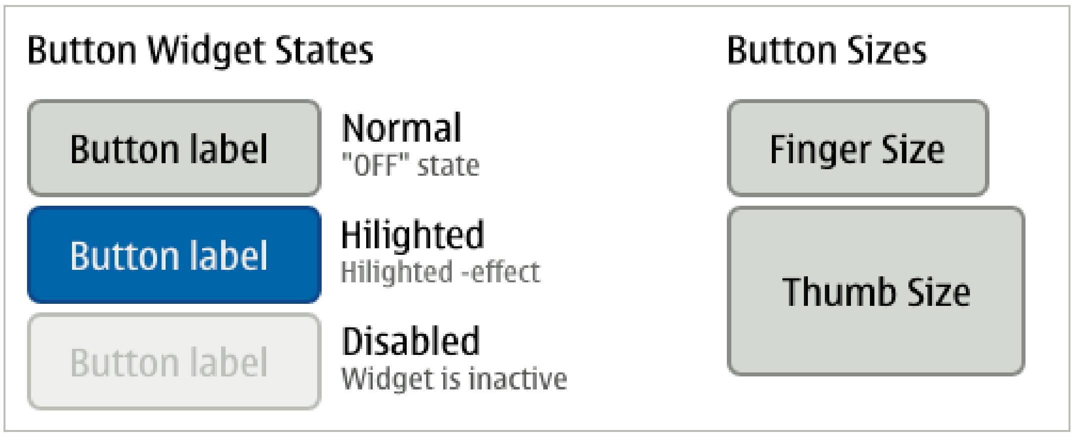


Figure 24: Finger Button and Thumb Button

| Area | Event | Action |
|---|---|---|
| Button | Finger press | Show highlight effect |
| Button | Finger Release | Invoke "clicked" callback function |
| Button | Dragging motion | Remove the highlight effect |

Table 12: Touch Button actions

### 13.1.1 Structure

| Property | Type | Defaults |
|---|---|---|
| Label | Text string | Empty |
| Sensitive (accepts input) | Yes/No | Yes |

Table 13: Touch Button properties

**Note:** Button label can also be replaced with any GTK widget or combination of GTK widgets. See *Hildon 2.2 UI Style Guide* for guidance for recommended and allowed styles.

### 13.1.2 Implementation

Implemented with the `GtkButton` class using `hildon_gtk_button_new()`.

To create a special button that wraps a longer string in two lines, use `hildon_gtk_button_new()`, set the font size to `SmallSystemFont` and define the string as `"Longer\ntext"`. See the figure below.

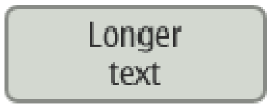


Figure 25: Button with longer text

## 13.2 Aligning and theming complex button text

The views and dialogues can have buttons with two text strings. This is particularly used in Pickers (the picker button), but it is used in normal buttons; for example, when an application-specific, custom picker-like UI is created.

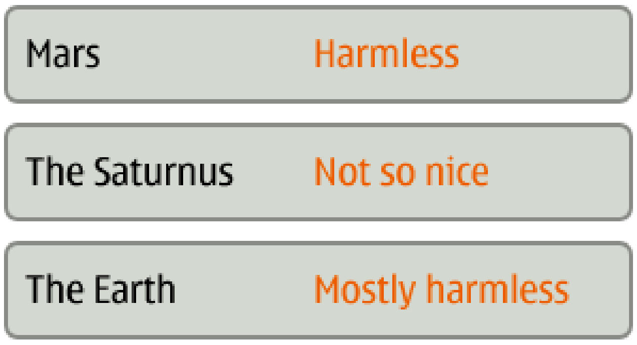


Figure 26: Complex button text with single line and two columns

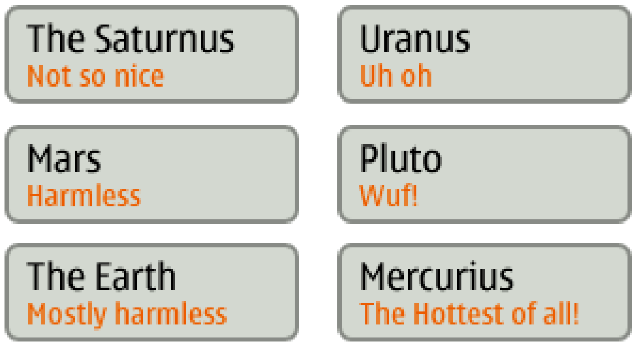


Figure 27: Complex button text with two lines

There are two main cases for aligning, of which the single-line case is the most common:

- Two text strings are in a single line on two "columns", both left aligned to the edge of its column
- Two text strings are in two rows, both left aligned to its row

There is only one main case for theming the two text strings (at least this is the current understanding).

- The first text string is "title" and themed with the default text theme. The second text is "value" and themed with a special font style for values, possibly similar to highlight style.

#### 13.2.1 Implementation

Implemented with the `HildonButton` class. When `HildonButton` is used to create a custom, picker-like button, set the correct theming with `hildon_button_set_style()`.

For a group of buttons, the column alignment is done with `GtkSizeGroup` for the `GtkLabels` of the "title" texts.

### 13.3 Hildon Touch Checkbox

Hildon Touch Checkbox is an "ON/OFF" toggle control, designed for finger use with touch screen input.

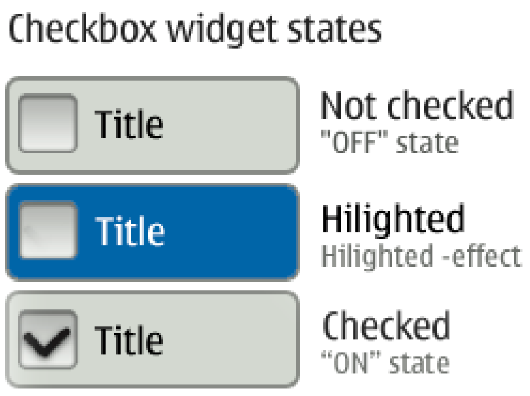


Figure 28: Checkbox states

The Checkbox widget should only be used for an individual setting that has a single on/off value. For example, "alarm active = on/off".

For other use cases, see Section 13.4, ‘Hildon Touch Toggle button’.

| Area | Event | Action |
|---|---|---|
| Button | Finger press | Highlight effect is shown |
| Button | Finger Release | Switch button state from ‘checked’ to ‘not checked’ or vice versa. Change the value of the button accordingly. |
| Button | Dragging motion | Highlight effect is removed |

Table 14: Touch Checkbox actions

#### 13.3.1 Structure

See Section 13.1, ‘Hildon Touch Finger button, Thumb button’.

Additionally, on top of the button, a check box (checked or not checked, depending on the state) is drawn, aligned to the left edge of the button. The button title is aligned left, and placed on the right side of the check box.

### 13.3.2 Implementation

Implemented with the `HildonCheckButton` class.

## 13.4 Hildon Touch Toggle button

Hildon Touch Toggle Button is an "ON/OFF" toggle control, designed for finger use with touch screen input.

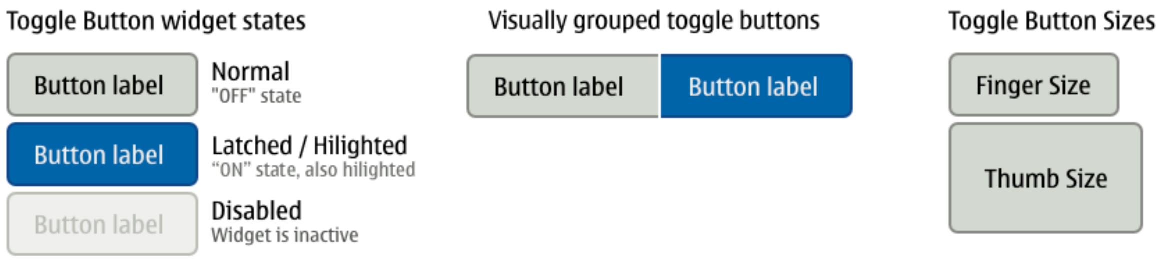


Figure 29: Toggle button states

The toggle button should only be used, if other widgets do not provide the desired functionality:

- For a single setting, the recommended widget is Hildon Touch Checkbox.
- For a group of values that work like radio buttons:

*Figure 12:* For 'embedded use', the recommended widget is Hildon Touch List in single selection mode.

Figure 13: For indirect use, the recommended widget is Hildon Touch List Picker.

- For achieving similar functionality as a group of check boxes:

*Figure 14:* For 'embedded use', the recommended widget is Hildon Touch List in multiple selection mode.

*Figure 15:* For indirect use, the recommended widget is Hildon Touch List Picker with multiple selection.

- *Only* if a free form layout is necessary for a group of values, then Hildon Touch Toggle Button can be used. For example, if a two column and three row group of buttons is needed.

| Area | Event | Action |
|---|---|---|
| Button | Finger press | Highlight effect is shown. |
| Button | Finger Release | Switch button state from highlight to non-highlight or vice versa. Change the value of the button accordingly. |
| Button | Dragging motion | Remove the highlight effect, only if non-highlight state is set. |

Table 15: Touch toggle button actions

### 13.4.1 Structure

See Section 13.1, 'Hildon Touch Finger button, Thumb button'.

Additionally a group of toggle buttons can be visually joined together, forming a "toggle group", as used for example in filters in Hildon Touch View Menu. When this visual style is used, the toggle button group must behave like radio buttons, that is, one and always only one of the buttons is highlighted.

### 13.4.2 Implementation

Implemented with the `GtkToggleButton` class using `hildon_gtk_toggle_button_new()`.

Visually grouped toggle buttons are packed inside a `GtkHBox` with `homogeneous` set to TRUE and `spacing` set to 0.

# 14 Data controllers

This section lists all data controllers, used to change the values of a single data object. These are typically more complex widgets, based on the simple ones.

**Implementation**

A general data controller is implemented as a base for the next widgets. It is implemented on the widget `HildonTouchSelector`. A current implementation and examples are provided.

## 14.1 Hildon Touch Date

Embeddable widget that contains three 'columns' of values: day, month, and year.

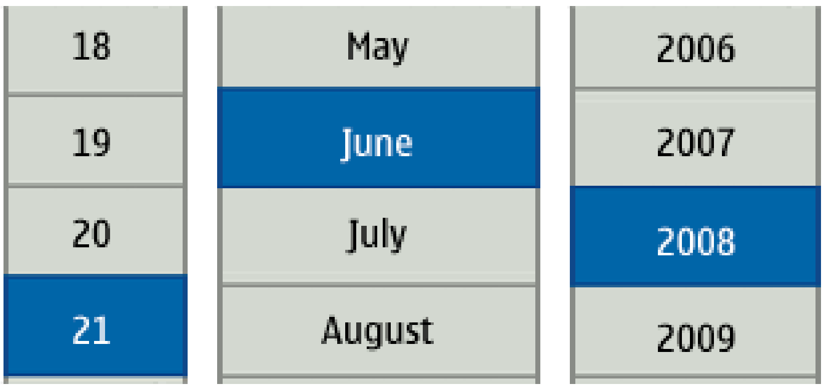


Figure 30: Touch Date

### 14.1.1 Structure

- FIRST COLUMN: Hildon Touch List in single selection mode, list values formatted with `wdgt_va_day_numeric` (1-31, varies for each month).
- SECOND COLUMN: Hildon Touch List in single selection mode, list values formatted with `wdgt_va_month` (months from January to December)
- THIRD COLUMN: Hildon Touch List in single selection mode, list values formatted with `wdgt_va_year`. The list items include current year, 100 years to the past (above the current year) and 50 years into the future (after the current year).

When the month selection is changed, the number of days displayed might change. If the new selected month does not have enough days to show the currently selected date, the last day of the month is selected. The same thing happens after the year selection is changed when February 29th has been selected in a leap year.

**Note:** The order of the columns depends on the (UNIX) locale used. For example, for the EN-US locale, the order is 'Month / Day / Year'. The user can change the locale indirectly by changing the Regional settings in Language and regional settings Control Panel Applet (CPA).

### 14.1.2 Implementation

This data controller is implemented on the widget `HildonDateSelector`. A current implementation and examples are provided.

For custom widgets handling date (for example correct column ordering needed), see `_locales_init()`, `_init_column_order()` and others in `hildon-date-selector.c`.

## 14.2 Hildon Touch Time

Embeddable widget that contains two or three 'columns' of values: hours, minutes, am/pm.

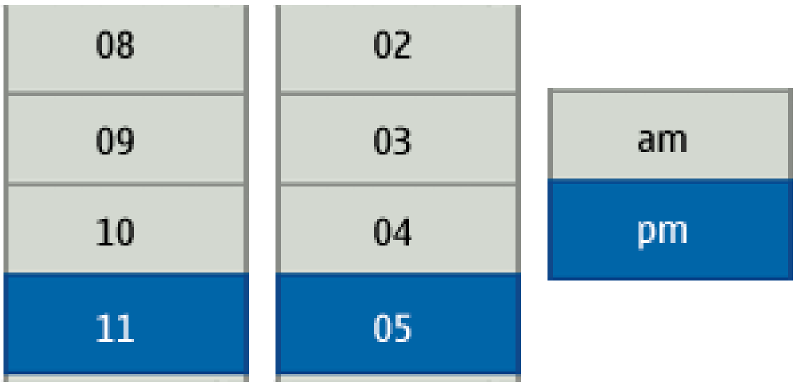


Figure 31: Touch Time

### 14.2.1 Structure

- FIRST COLUMN: Hildon Touch List in single selection mode, list values formatted with `wdgt_va_24h_hours` (00-23) or `wdgt_va_12h_hours` (12,01,02,...,11), depending on 24h/12h setting.
- SECOND COLUMN: Hildon Touch List in single selection mode, list values formatted with `wdgt_va_minutes` (00-59, 1 minute stepping).
- THIRD COLUMN: optional, shown depending on 24h/12h setting. Hildon Touch List in single selection mode, list values formatted with `wdgt_va_am` and `wdgt_va_pm`.

The 24h/12h display format is set in Date and Time CPA (Control Panel Applet). See Section 18.4, 'Time format strings', for more information.

### 14.2.2 Implementation

This data controller is implemented on the widget `HildonTimeSelector`. A current implementation and examples are provided.

## 15 Pickers

This section lists all picker widgets. The pickers are the way to do indirect value selection. For example, in 'edit alarm' dialogue, a Time Picker can be used to select the time.

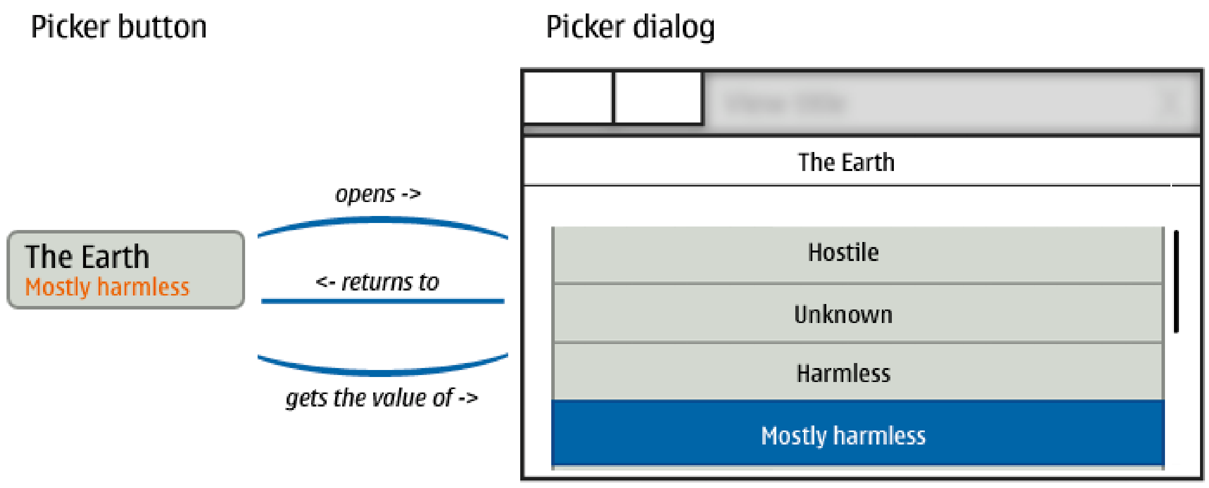


Figure 32: Picker widget diagram

The widget consists of two parts - a 'picker button' that both shows setting title (that is, picker title) and the currently selected (or initially default) value in itself, and, when tapped, represents a list of choices to the user in a dialogue. The user makes a choice and dismisses the dialogue. The picker button then shows the newly selected value.

**Note:** Picker widget supports multiple columns. See, for example, Date picker and Time picker.

**Note:** The content (title + value) of the picker button is not necessarily on two rows. It can also be on a single row, with title on the left and value on the right. The single row style is considered as the more typical case.

Common structure for the pickers:

- PICKER BUTTON

Figure 16: picker title

Figure 17: button value (can be either in second row, below title, or on a single row, right of the title)

- PICKER DIALOGUE

Figure 18: The dialogue title shows the picker title.

Figure 19: Data controllers of each picker.

Figure 20: The button in the dialogue is `wdgt_bd_done`, which closes the dialogue and returns the new value chosen by the user.

- Picker values (data formats): In the UI, the following three data formats are allowed:

Figure 21: single string

Figure 22: single icon

Figure 23: single icon (left) + single string (right of icon)

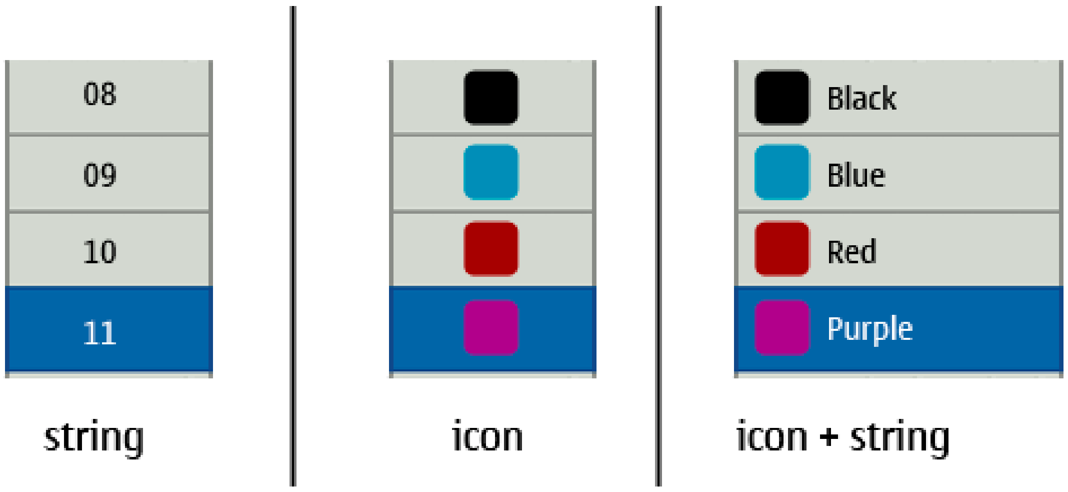


Figure 33: Picket values (data formats)

### 15.1.1 Implementation

The more general widgets that implement this behaviour are `HildonPickerDialog` and `HildonPickerButton` that manages the dialogue and button behaviour, and a `HildonTouchSelector` widget that manages the data (it is the data controller).

**Note:** The implementation of `HildonTouchSelector` supports multiple columns. This is because it is the parent class of `HildonDateSelector` and `HildonTimeSelector`, which require multiple columns. This can be a little contradictory with the previously said 'Picker widget does NOT support multiple columns, except for Date picker and Time picker', but it was made to simplify the implementation process.

## 15.2 Hildon Touch List picker

Also known as Hildon Touch List Selector, this is a finger-usable widget for selecting an item from a predefined list.

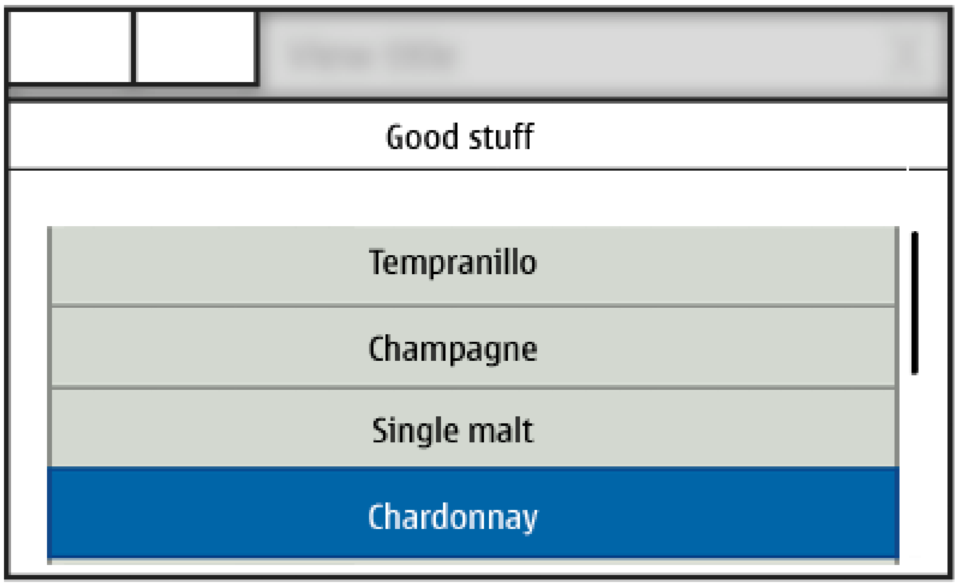


Figure 34: Touch List picker

### 15.2.1 Structure

- DIALOGUE CONTENT: Hildon Touch List in single selection mode.

As an exception to all other List pickers, this simple List picker does not have 'Done' button in the picker dialogue, but instead whole width of the dialogue is used for content. This means that tapping on an item in the list also closes the picker dialogue and changes the value.

### 15.2.2 Implementation

This is implemented with `HildonPickerDialog`.

The most common use case is using that with the picker button, as the picker diagram shows. This can be implemented directly by using a `HildonPickerButton`. Internally this widget manages to create the dialogue (`HildonPickerDialog`), and the `HildonPickerButton` API has methods to create the list in an easy way. For more complex use cases (such as lists with a renderer different to the text), you can directly access the data controller (`HildonTouchSelector`) included in the dialogue.

Examples for using this widget exist in the `libhildon1-doc / libhildon1-examples` packages.

## 15.3 Hildon Touch List picker with multiple selection

Also known as Hildon Touch List Selector with multiple selection. This is a finger-usable widget for selecting one or several items from three predefined lists.

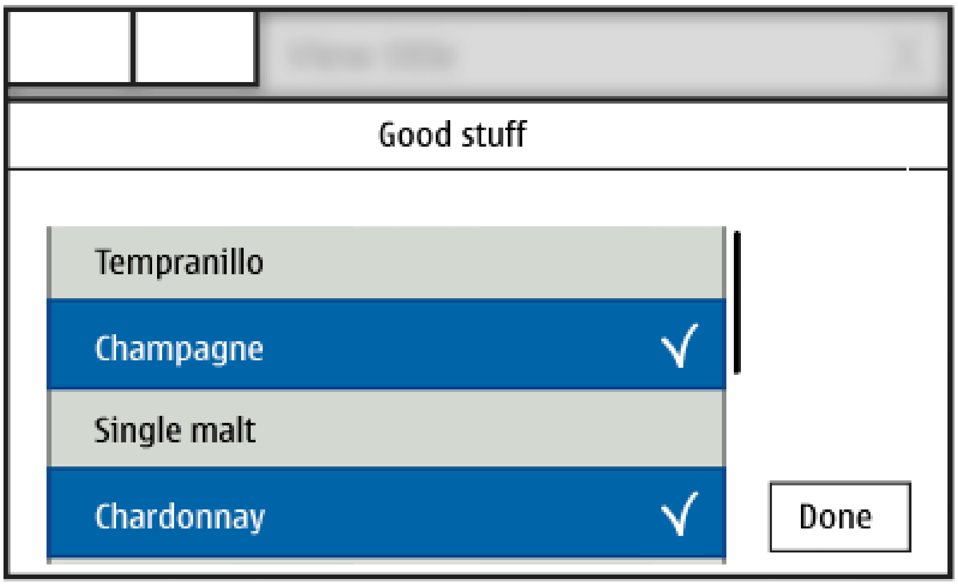


Figure 35: Touch List picker with multiple selection

### 15.3.1 Structure

- DIALOGUE CONTENT: one column with Hildon Touch List in multiple selection mode

### 15.3.2 Implementation

See the single selection mode picker. To set the multiple selection mode, use the following:

```
hildon_touch_selector_set_column_selection_mode(...,
HILDON_TOUCH_SELECTOR_SELECTION_MODE_MULTIPLE);
```

## 15.4 Hildon Touch List picker with entry

Also known as Hildon Touch List Selector with entry, this is a finger-usable widget for selecting one value from predefined lists or inputting a value via text entry.

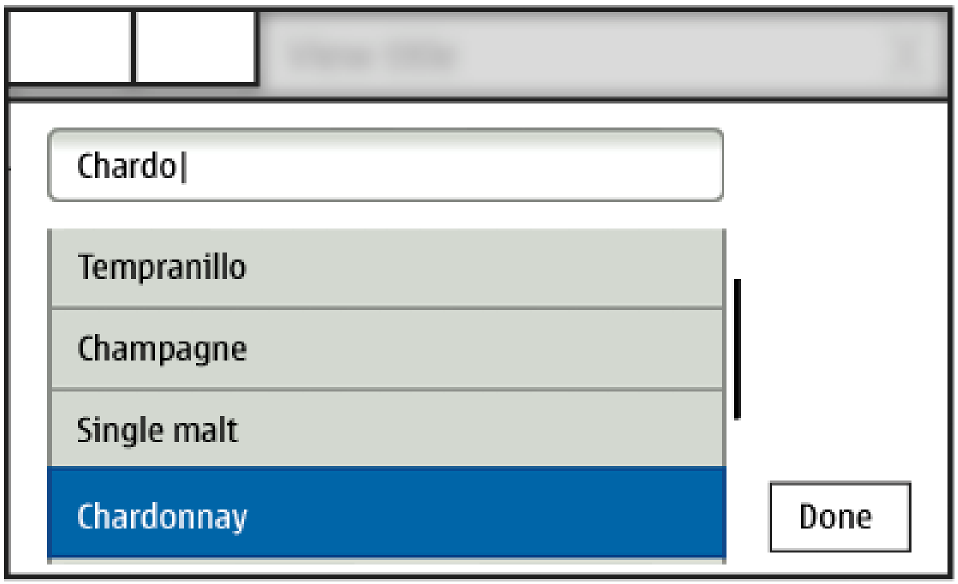


Figure 36: Touch List picker with entry

### 15.4.1 Structure

- DIALOGUE CONTENT, vertically from top to bottom:

Figure 24: Hildon Touch Text Entry. The text in this input field affects the highlight of the list below. The functionality is similar to a combo box.

Figure 25: Hildon Touch List in single selection mode. Note that the text in the list items is left-aligned.

### 15.4.2 Implementation

See the single selection mode picker. To use picker with entry, create `hildon_touch_selector_entry_new()` and use that with the picker button or dialogue.

## 15.5 Hildon Touch Date picker

This widget is the picker variant of the Hildon Touch Date widget.

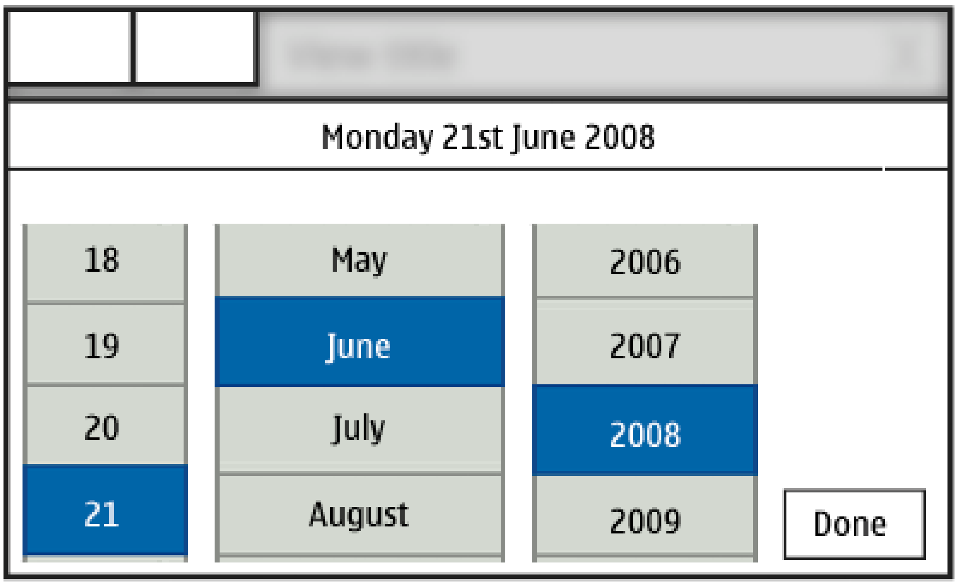


Figure 37: Touch Date picker

### 15.5.1 Structure

- DIALOGUE CONTENT: **Hildon Touch Date**

### 15.5.2 Implementation

This is implemented on the widgets `HildonDateButton` and `HildonDateSelector` (the data controller).

The most common use case is to use only `HildonDateButton` as the picker diagram shows. This widget creates internally the `HildonPickerDialog` and the `HildonDateSelector` data controller. The data controller can also be used without the dialogue.

Example of this exists in the `libhildon1-doc / libhildon1-examples` packages.

## 15.6 Hildon Touch Time picker

This widget is the picker variant of the Hildon Touch Time widget.

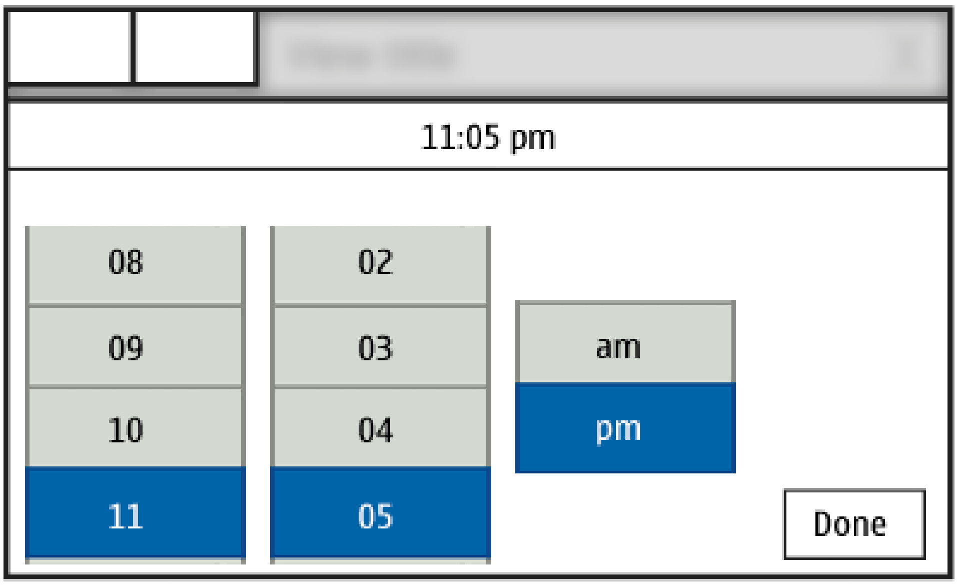


Figure 38: Touch Time picker

### 15.6.1 Structure

- DIALOGUE CONTENT: Hildon Touch Time

### 15.6.2 Implementation

This is implemented on the widgets `HildonTimeButton` and `HildonTimeSelector` (the data controller).

The most common use case is to use only `HildonTimeButton` as the picker diagram shows. This widget creates internally the `HildonTimeDialog` and the `HildonTimeSelector` data controller. The data controller can also be used without the dialogue.

Example of this exists in the `libhildon1-doc / libhildon1-examples` packages.

# 16 Text editors

This chapter lists all text editor widgets. These widgets are the way for the user to input text with HW keyboard or virtual keyboard.

## 16.1 Common features

| Area | Event | Action |
|---|---|---|
| Widget | Finger press | Widget gets input focus, highlight and cursor is shown.<br>Input field background changes to strong white (handled by theme). |
| Widget | Finger Release | - |
| Widget | Keyboard events on focused Widget | Content is edited accordingly. |
| Outside Widget | Finger press | If widget has input focus, the focus is removed.<br>Input field background changes to slightly less white, yet not inactive colour (handled by theme). |

Table 16: Text editor widget actions

| Property | Type | Defaults |
|---|---|---|
| Content | Text string | Empty |
| Focus state | ON/OFF | OFF |
| Sensitive (accepts input) | Yes/No | Yes |
| Password mode ("*****", only used in Hildon Touch Text Entry) | ON/OFF | OFF |

Table 17: Text editor widget properties

## 16.2 Placeholder text

Hildon Touch Text Entry and Hildon Touch Text Area can be used without a title text, if the following conditions are met:

- Only one text input field in a dialogue can be without title. Other input fields must have a title.
- The no-title field should be the field that describes the object or task that is being performed. For example, "Event title" field for the Edit event dialogue.
- The no-title field should be the first field in the dialogue, as long as it makes sense in the layout.
- The no-title field must have a placeholder text inside the text input field, to indicate what the field is about.
- When text input field is tapped (that is, when it gets focus), the placeholder text is removed from the input field.

- The placeholder text is never stored as a value of the input field. If no text is written to the field by the user, the value of the field is empty string. Optionally, there can be the additional requirement that something must be written to the field, before saving is possible.

```
[ Phonenumber ] - Empty entry (caption shown in SecondaryTextColour)
[ | ] - Entry focused (caption hidden)
[ +3580400| ] - Value written (value is shown with default text color)
[ +3580400555555 ] - Entry has value and it is not focused. Text shown
with normal color.
```

Example 4: Placeholder text, entry asking phone number

## 16.3 Hildon Touch Text Entry

Also known as Hildon Touch Text Editor, this widget allows the user to enter one line of text. The widget accepts input focus by clicking, which is shown with a visible, blinking text cursor and a highlight effect around the widget. When the widget is focused, the input field background is stronger white than without focus. The behaviour of the widget is similar to Hildon 2.1.

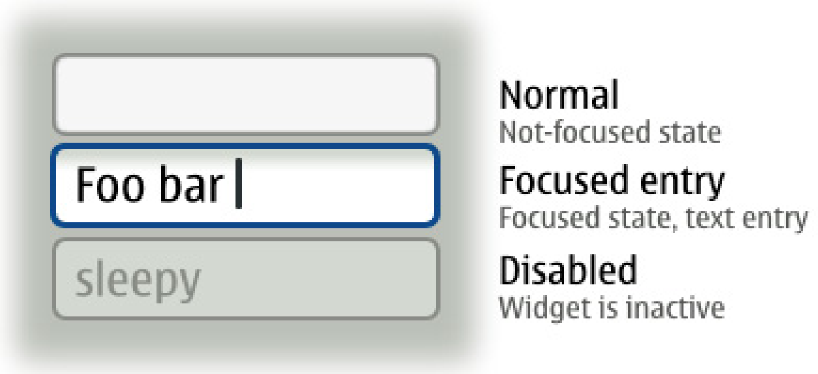


Figure 39: Touch text Entry with widget states

### 16.3.1 Implementation

Implemented using the `HildonEntry` class, with `hildon_entry_get_text()` and `hildon_entry_set_text()`.

To set the placeholder text, use `hildon_entry_set_placeholder()`.

To set a specific input mode, use `g_object_set (G_OBJECT (entry), "hildon-input-mode", HILDON_GTK_INPUT_MODE_NUMERIC, NULL);` or similar.

## 16.4 Hildon Touch Text Area

Also known as Hildon Touch Text View, this widget is similar in functionality and appearance to the text entry widget, the difference being that it can handle multiple lines of text. When the widget is focused, the input field background is stronger white than without focus.

The behaviour of the widget is different to Hildon 2.1 in following ways:

- The widget is noted with scrollbars.
- The widget should always automatically resize (in height) when more text is entered.
- Finger usage is not supported for moving the cursor or selecting the text. Finger movement is reserved for the pannable area inside the view or dialogue.

- Text selection is only supported by using the HW keys (pressing SHIFT and moving the arrow keys).
- Cut/Copy/Paste is only handled from the HW keys (CTRL-C etc.) and from the full-screen virtual keyboard menu.

The cursor in the text field should be thicker than 1 pixel wide, to make it more usable on very small screens with large DPI.

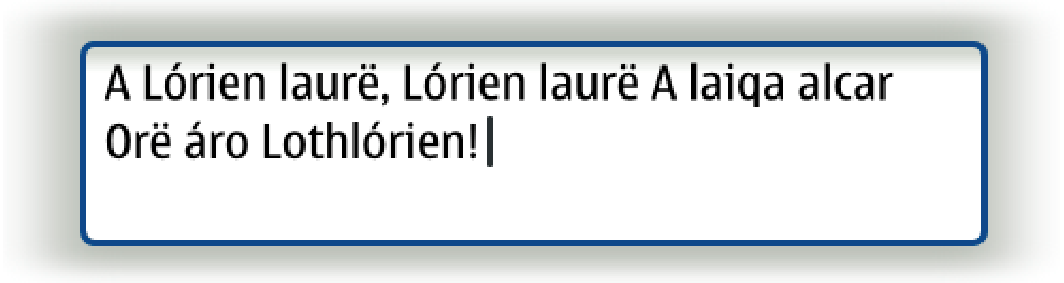

A Lórien laurë, Lórien laurë A laiqa alcar
Orë áro Lothlórien!

Figure 40: Touch Text Area with focus and entered text

### 16.4.1 Implementation

Hildon Touch Text Area is implemented using the `HildonTextView` class. Always use Hildon Touch Pannable Area inside the dialogue or view when using this widget.

To set the placeholder text, use `hildon_text_view_set_placeholder()`.

# 17 Progress indicators and progress controllers

This chapter lists all widgets related to indicating and controlling progress.

## 17.1 Hildon Touch progress indicator

In Hildon 2.1, the progress indication was placed 1) in toolbar as progress bar, 2) as progress animation, inside information banner, or 3) as a Cancel note.

In Hildon Touch widgets, the recommended way to use progress indication is to show a progress animation directly on the UI. This means showing a progress animation next to the title. For views, the progress animation should be shown in the view title (in title area), and for dialogues, the animation should be shown in the dialogue title. See the Figures below for more information.

It is recommended that the indicator is shown only if processing is taking more than 1,5 seconds.

**Note:** If you want to be able to stop the process, use Hildon Touch Progress Note instead.

**Note:** For performance reasons it is acceptable to have static progress image `widgets_progress_indicator_static` for indicating progress inside a content item (see Figure 43). However, Progress indication in content should be used only when progress indication is not sensible in view title or dialogue title.

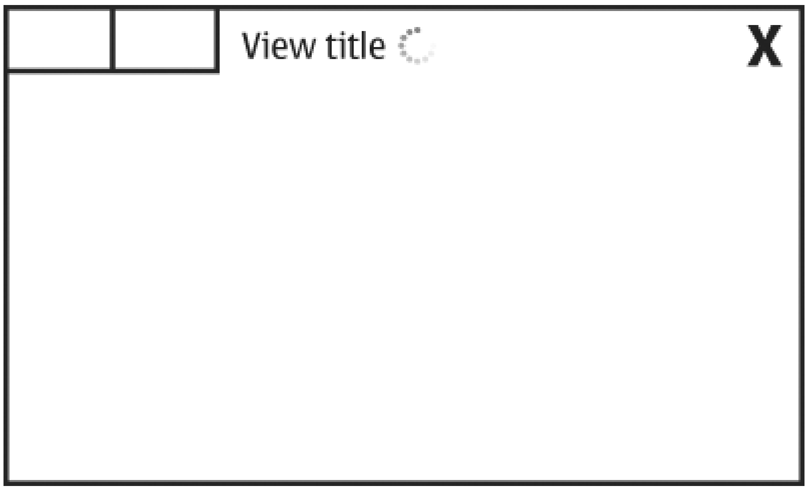


Figure 41: Progress indication in a view

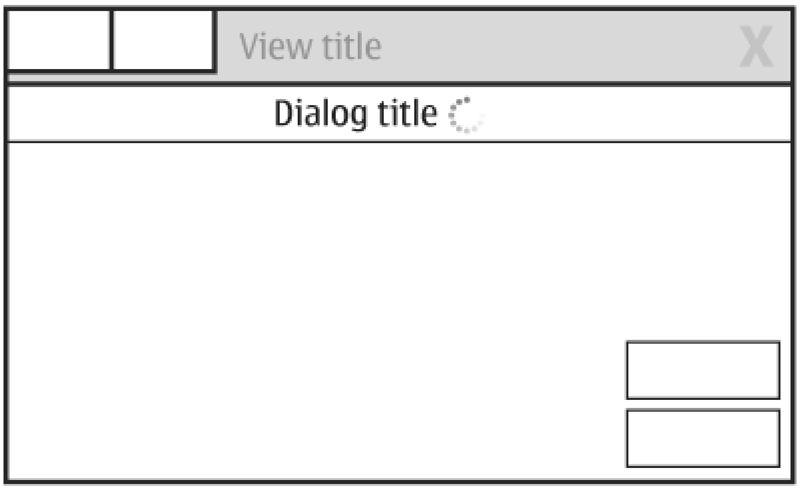


Figure 42: Progress indication in a dialogue

Figure 43: Progress indication inside content

**Note:** The progress indication must be visible, when it is enabled. The Title text must be truncated if needed, to make this happen.

More verbose progress indication is not encouraged, but if text needs to be shown, use normal Hildon Touch Information banner (fades away automatically) to inform the user about what is in progress. The progress indication itself remains for the whole duration of the process.

#### 17.1.1 Implementation

The progress indicator is set visible and hidden with `hildon_gtk_window_set_progress_indicator()`.

### 17.2 Hildon Touch Progress Note

Hildon Touch Progress Note is a widget for showing progress (also known as Cancel Note), while allowing the user to stop the process at any time. This should be used when a time-consuming event is happening and the user should wait.

**Note:** Touch Progress Note cannot be closed by tapping on the dimmed area.

**Note:** You should only use this widget when the ability to stop the process is absolutely necessary. Otherwise use Hildon Touch Progress Indicator instead.

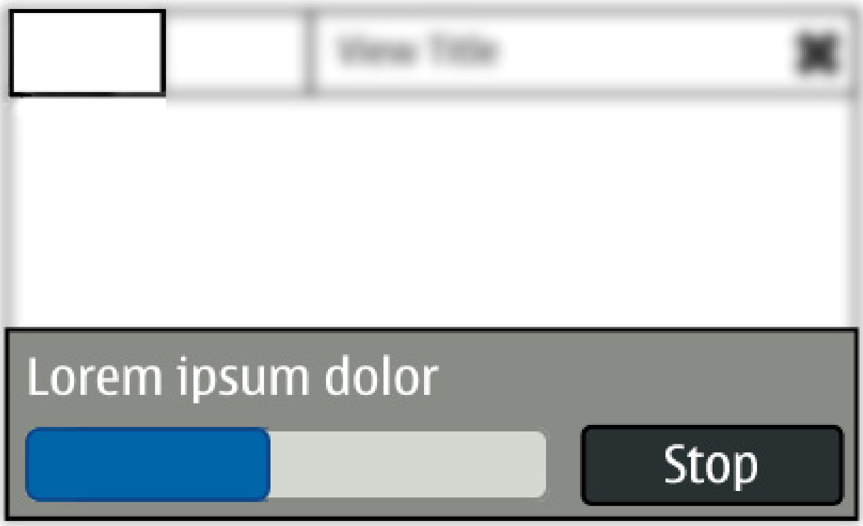


Figure 44: Touch Progress Note

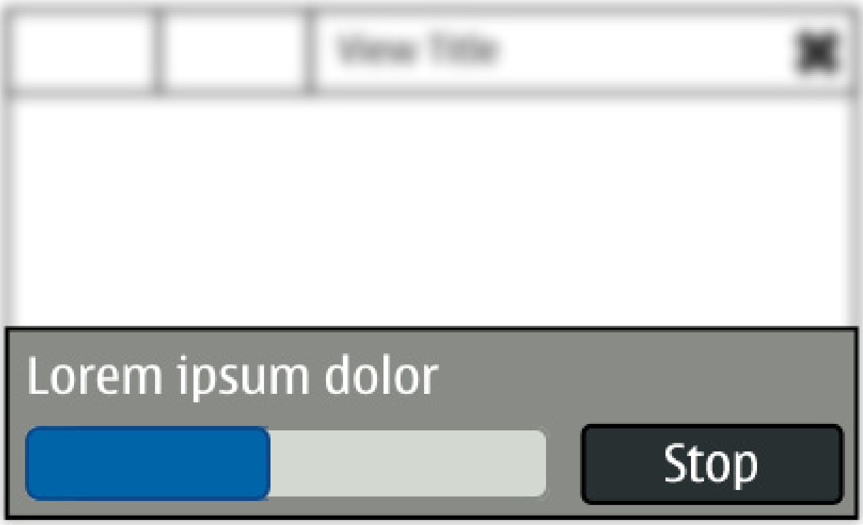


Figure 45: Touch Progress Note with system modal dialogue

### 17.2.1 Implementation

The progress and cancel notes are created with `hildon_note_new_cancel_with_progress_bar()`.

## 17.3 Hildon Touch Progress Bar

Hildon Touch Progress Bar is a widget to show the user that a time-consuming event is happening and the user should wait, for example an image is being resized, or some content loaded over the network connection.

When possible, Hildon Touch Progress Indicator should be used instead of the progress bar.

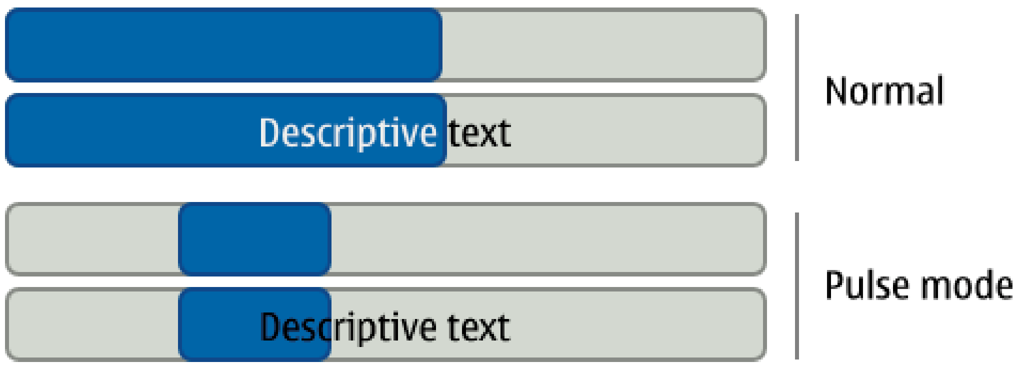


Figure 46: Touch Progress Bar

Progress bar does not support any user interaction, it exists only to display progress information.

**Note:** The height of the progress bar is customisable, so it is possible to specify a bar with half of the normal height. This can be used, for example, when there is a need to have a second row in a button.

### 17.3.1 Structure

| Property | Type | Defaults |
|---|---|---|
| Descriptive text label | Text string | Empty |
| Minimum value | Numeric | 0 |
| Maximum value | Numeric | 1? |
| Progress value | Numeric, between min and max values | 0 |

Table 18: Touch Progress Bar widget properties

**Note:** Progress Bar does NOT have a disabled state.

### 17.3.2 Implementation

Hildon Touch Progress Bar is implemented using the `GtkProgressBar` class. Behaviour is same as in Hildon 2.1.

## 17.4 Hildon Touch Seek Bar

Hildon Touch Seek Bar lets the user select a value from a range of predetermined values by adjusting the slider "thumb" handle with a finger drag. A larger version of the Hildon Seek Bar is implemented with theming. The behaviour is otherwise the same as before, except that the plus and minus buttons are removed from the endpoints since the widget itself is large enough to be directly finger-adjustable.

For a horizontal seek bar, the left side of the bar up to the handle is coloured with the highlight colour (see the image below).

For a vertical seek bar, the bottom side of the bar up to the handle is coloured with the highlight colour.

**Note:** This widget does not support showing numeric values. Use separate label widgets for that purpose.

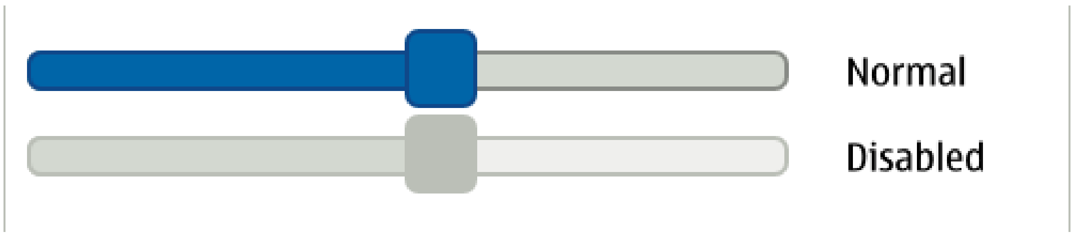


Figure 47: Touch Seek Bar

| Area | Event | Action |
|---|---|---|
| Widget | Finger press | "Thumb handle" is moved to the pressed location and thus the value is set |
| "Thumb" handle | Dragging motion | "Thumb" handle is moved to the dragged location |

Table 19: Touch Seek Bar widget actions

### 17.4.1 Structure

| Property | Type | Defaults |
|---|---|---|
| Minimum value | Numeric | 0 |
| Maximum value | Numeric | 1? |
| Set value | Numeric, between min and max values | 0 |
| Sensitive (accepts input) | Yes/No | Yes |

Table 20: Touch Seek Bar widget properties

### 17.4.2 Implementation

Implemented using the `GtkHScale` and `GtkHVScale` classes. `hildon_gtk_hscale_new()` must be used to get the proper finger interaction behaviour. For indicating the current (numeric) value(s) of the Seek bar, use separate `GtkLabel`(s).

### 17.4.3 Implementation for custom negative-positive Seek Bar

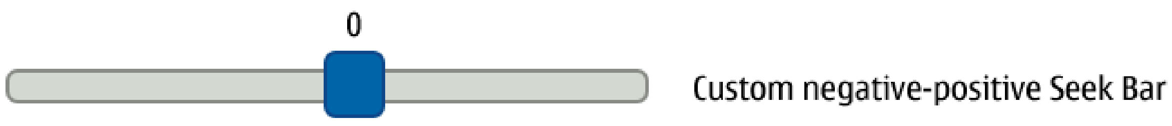


Figure 48: Custom Seek bar

When the seek bar is used with values ranging from negative to positive (for example -2..0..2), the "filling effect" must be removed from the widget. This can be done by setting the widget name of the `GtkHScale` to "bidirectional-slider-horizontal" or `GtkVScale` to "bidirectional-slider-vertical". The value label on the upside of the widget is created with standard `GtkLabel`.

# 18 Other UI notes

## 18.1 Icons used by widgets

| Icon | Icon ID | Description |
|---|---|---|
| ❐ | <defined in theme template> | Tasks button icon for Task Switcher (the UI Framework) |
| ▦ | <defined in theme template> | Tasks button icon for Task Launcher (the UI Framework) |
| ⬅ | <defined in theme template> | Back button of the UI Framework |
| ✖ | <defined in theme template> | Close button of the UI Framework |
| ✓ | widgets_tickmark_list | Generic tick mark for list items |
| ✓ | widgets_tickmark_grid | Generic tic mark for grid items |
| ◌ | <defined in theme template> | Progress indication animation |
| ◌ | widgets_progress_indicator_static | Static progress indication |
| ▭ | general_fullsize | Toolbar button for fullscreen (full colour) |

| Icon | Icon ID | Description |
|---|---|---|
| ⬅ | general_overlay_back | Overlay (back-)button for fullscreen (1bit colour) |

Table 21: Widget icons and their descriptions

## 18.2 Common dialogue button labels

In accordance to Hildon 2.2 UI Style Guide, the button commands should always describe an *action*, not a statement. Button labels like OK and cancel should NOT be used.

Guidelines for naming:

- For settings-like dialogue, containing many settable settings, use `wdgt_bd_save`.
- In dialogues used for saving a new value for the first time, use `wdgt_bd_save`.
- For a dialogue that sets value(s) for just ONE setting, use `wdgt_bd_done`.
- If in doubt, try `wdgt_bd_done`. If it does not work, choose a more appropriate action.
- NEVER use remove. Use `wdgt_bd_delete` instead, to align with Nokia terms. Same goes for any other localisable strings too.
- Use `wdgt_bd_new` when opening a wizard, creating a new object (for example an account), or creating a new content item.
- Use `wdgt_bd_edit` when editing or changing something created with the "new" button, or when editing or changing a content item.
- Use `wdgt_bd_view` when viewing details of an object or content item.

| Description | Logical name | UI string |
|---|---|---|
| Button label | wdgt_bd_done | "Done" |
| Button label | wdgt_bd_save | "Save" |
| Button label | wdgt_bd_yes | "Yes" |
| Button label | wdgt_bd_no | "No" |
| Button label<br>Note that for feminine congrugation (localisation) of the "new" string in latin-based languages, applications MUST define their own string. | wdgt_bd_new | "New" |
| Button label | wdgt_bd_edit | "Edit" |
| Button label | wdgt_bd_move | "Move" |
| Button label | wdgt_bd_delete | "Delete" |
| Button label | wdgt_bd_add | "Add" |
| Button label | wdgt_bd_view | "View" |
| Button label | wdgt_bd_search | "Search" |
| Button label | wdgt_bd_sort | "Sort" |
| Button label | wdgt_bd_share | "Share" |

| Description | Logical name | UI string |
|---|---|---|
| Button label | wdgt_bd_rename | "Rename" |
| Button label | wdgt_bd_stop | "Stop" |
| Button label.<br>Using this is not recommended. However, if it is absolutely necessary to have a button for just closing a dialogue, use this string. | wdgt_bd_back | "Back" |

Table 22: Button labels

Note that:

- “Close" button string is DEPRECATED. There should be no need for this button, since the dialogue can be closed by tapping outside it.
- "Ok" button string is DEPRECATED. It should NOT be used in the UI. Use a word describing an action instead. Use `wdgt_bd_done`, if unsure.
- "Cancel" button string is DEPRECATED. The dialogues are cancelled by tapping the dimmed area.

## 18.3 Date format strings

**Note:** In lists that show the date or time of items, the recommended style is to
1) show only time (`wdgt_va_24h_time` or `wdgt_va_12h_time`) for items that have today's time, and
2) show only date for items that do not have today's time (`wdgt_va_date_day_name_short` or `wdgt_va_date`).

```
Message one     15:00          Message one    15:00
Message two     13:00          Message two    13:00
Message three   12.3.2008      Message three  Mon 12.3.
Message four    14.3.2008      Message four   Wed 14.3.
```

Example 5: Date format strings

Table 23: Date format strings

| Description | Logical name | UI string |
|---|---|---|
| Localised long format date. The default date format is "XXXXXXX dd. mmmmmmm yyyy", where XXXXXXX is the name of the weekday, dd is the number of the day, mmmmmmm is the full name of the month and yyyy is the full number of the year, for example "Monday 21. June 2008".<br>Use this logical string with the `strftime` function. | wdgt_va_date_long | "%A %e. %B %Y" |

| Description | Logical name | UI string |
|---|---|---|
| Localised date with the day name in shortened form (for example, "Thu 31.03."). The date format is "XXX dd.mm.", where XXX is the up to three first characters of the weekday, dd is the number of the day, and mm is the number of the month.<br>Use this logical string with the `strftime` function. | wdgt_va_date_day_name_short | "%a %d.%m." |
| Localised full date with the day name in shortened form (for example, "Thu 31.03.08"). The date format is "XXX dd.mm.yy", where XXX is the up to three first characters of the weekday, dd is the number of the day, mm the number of the month, and yy is the short form of year.<br>Use this logical string with the `strftime` function. | wdgt_va_fulldate_day_name_short | "%a %d.%m.%y" |
| Localised full date (for example, "21. June 2008"). The date format is "dd. mmmmmm yyyy", where dd is the number of the day, mmmmmm is the full name of the month and yyyy is the full number of the year.<br>Use this logical string with the `strftime` function. | wdgt_va_date_medium | "%e. %B %Y" |
| The localised numeric form of the date. For example, "12.03.2008" for 12th March 2008.<br>Use this logical string with the `strftime` function. | wdgt_va_date | "%d.%m.%Y" |
| The localised short numeric form of the date without the year. For example, "12.03." for 12th March 2008.<br>Use this logical string with the `strftime` function. | wdgt_va_date_short | "%d.%m." |
| Month and year date format. For example, "January 2008".<br>Use this logical string with the `strftime` function. | wdgt_va_fullmonth_year | "%B %Y" |
| Localised full numeric form of a year. For example, "2008".<br>Use this logical string with the `strftime` function. | wdgt_va_year | "%Y" |

| Description | Logical name | UI string |
|---|---|---|
| Localised full name of a month. For example, "January".<br>Use this logical string with the `strftime` function. | wdgt_va_month | "%B" |
| The localised short form of the name for month. For example, "Mar" for March.<br>Use this logical string with the `strftime` function. | wdgt_va_month_name_short | "%b" |
| The localised full name for a weekday. For example, "Wednesday".<br>Use this logical string with the `strftime` function. | wdgt_va_week | "%A" |
| The localised short form of the name for a weekday. For example, "Wed" for Wednesday.<br>Use this logical string with the `strftime` function. | wdgt_va_week_name_short | "%a" |
| Week number (ISO 8601), for example "11" for the week 11.<br>Use this logical string with the `strftime` function. | wdgt_va_week_number | "%V" |
| Week number (ISO 8601) when Sunday is the first day of the week. For example, "11" for the week 11<br>Use this logical string with the `strftime` function. | wdgt_va_week_number_sunday_first | "%U" |
| Localised numeric form of a day.<br>Use this logical string with the `strftime` function. | wdgt_va_day_numeric | "%e" |

The output of these strings can be experimented with the date command in Linux tablet X Terminal, for example `date +"current date: %A %e. %B %Y"`. However, note that "date" does fully support locales. For details about `strftime` usage, see http://linux.die.net/man/3/strftime.

## 18.4 Time format strings

12h/24h implementation note:

In Fremantle, the 24h/12h setting is user-definable (Date and Time CPA) and not directly dependent on language/regional settings.

The current value for the setting can be read from `Gconf` property: `/apps/clock/time-format (TRUE - 24h; FALSE - 12h)`. Once this value is known, the appropriate string can be shown, for example `wdgt_va_24h_time` or `wdgt_va_12h_time_am` or `wdgt_va_12h_time_pm`.

To decide correctly between "am" or "pm", use `struct tm`:

```
if (tm->tm_hour > 11)
// wdgt_va_12h_time_pm
else
// wdgt_va_12h_time_am
```

| Description | Logical name | UI string |
|---|---|---|
| The localised 24h time. For example, "23:00".<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_24h_time | "%H:%M" |
| The localised 12h time, ante meridiem. For example, "11:00 am". Note: "am" needs to be localised as in S60.<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_12h_time_am | "%I:%M am" |
| The localised 12h time, post meridiem. For example, "11:00 pm". Note: "pm" needs to be localised as in S60.<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_12h_time_pm | "%I:%M pm" |
| The full localised 24h time, including seconds. For example, "23:00:04".<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_full_24h_time | "%T" |
| The full localised 12h time, including seconds. For example, "11:00:04 pm".<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_full_12h_time | "%r" |
| Localised hours in 24h clock.<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_24h_hours | "%H" |
| Localised hours in 12h clock.<br>Use this logical string with the `strftime` function. See 12h/24h implementation note for details. | wdgt_va_12h_hours | "%I" |
| Localised full numeric form of minutes and seconds.<br>Use this logical string with the `strftime` function. | wdgt_va_minutes_seconds | "%M:%S" |
| Localised full numeric form of minutes.<br>Use this logical string with the `strftime` function. | wdgt_va_minutes | "%M" |
| Localised full numeric form of seconds.<br>Use this logical string with the `strftime` function. | wdgt_va_seconds | "%S" |
| Localised name of ante meridiem. For example, "am". Note: this needs to be localised as in S60.<br>See 12h/24h implementation note for details. | wdgt_va_am | "am" |
| Localised name of post meridiem. For example, "pm". Note: this needs to be localised as in S60.<br>See 12h/24h implementation note for details. | wdgt_va_pm | "pm" |

Table 24: Time format strings

The output of these strings can be experimented with the date command in Linux tablet X Terminal, for example `date +"current time: %H:%M"`. However, note that "date" does fully support locales. For details about `strftime` usage, see http://linux.die.net/man/3/strftime.

## 18.5 Other platform UI banners and strings

| Description | Logical name | UI string |
|---|---|---|
| This OPTIONAL Information banner is shown when the hardware button + or - is pressed for adjusting the volume during a call, for example. Volume level is shown by numbers 1 - 100 (percentage). The change in volume level should be updated real-time in the banner. | wdgt_ib_volume | "Volume %d%%" |
| This OPTIONAL Information banner is shown when the hardware button + or - is pressed for adjusting the zoom level in, for example, the image viewer. Zoom level is shown by numbers (percentage). Valid values are defined by each application. The change in zoom level should be updated real-time in the banner. | wdgt_ib_zoom | "Zoom %d%%" |
| Title | wdgt_ti_date | "Date" |
| Title | wdgt_ti_time | "Time" |
| Plural_form 0: String for relative time of less than two minutes | wdgt_va_ago_one_minute | "One minute ago" |
| Plural_form 1: String for relative time of less than one hour | wdgt_va_ago_minutes | "%d minutes ago" |
| Plural_form 0: String for relative time of less than two hours | wdgt_va_ago_one_hour | "One hour ago" |
| Plural_form 1: String for relative time of less than one day | wdgt_va_ago_hours | "%d hours ago" |
| Plural_form 0: String for relative time of less than two days | wdgt_va_ago_one_day | "One day ago" |
| Plural_form 1: String for relative time of less than one year | wdgt_va_ago_days | "%d days ago" |
| Plural_form 0: String for relative time of less than two years | wdgt_va_ago_one_year | "One year ago" |
| Plural_form 1: String for relative time used otherwise (two or more years ago) | wdgt_va_ago_years | "%d years ago" |

Table 25: Other banners and strings

# 19 Deprecated widgets

This chapter lists all widgets that are no longer used in the Hildon 2.2 UI Style.

- Hildon 2.1 focus

Figure 26: Focus is not available or visible by default. Keyboard navigation is not supported by default.

Figure 27: For special modes, like single selection mode and multiple selection mode, highlight is supported. In such situations, however, activating the list items is NOT supported. See Chapter 3, ‘UI modes’, for more information.

- Split view (`GtkPaned`)

Figure 28: Use two separate **Hildon Touch Views**, taking advantage of the view / sub-view navigation

- Special/custom banners

Figure 29: Banners with icons are not supported.

Figure 30: Animation banner **-** Use Hildon Touch Progress indication instead, or Hildon Touch Progress Note.

Figure 31: Progress banner **-** Use Hildon Touch Progress indication instead, or Hildon Touch Progress Note.

- Tabbed dialogues (`GtkNotebook`)

Figure 32: Use Hildon Touch Pannable Area instead. This means using just one pannable dialogue.

- The Hildon 2.1 Fullscreen mode

Figure 33: Full screen is normally not used. No toggling between the modes.

Figure 34: For specific navigation flows, like viewing a single image, the fullscreen view is part of the normal navigation flow.

- Radio buttons

*Figure 35:* For "embedded use", the recommended widget is Hildon Touch List in single selection mode.

Figure 36: For indirect use, the recommended widget is Hildon Touch List Picker with multiple selection.

- Group of check boxes

*Figure 37:* For "embedded use", the recommended widget is Hildon Touch List in multiple selection mode.

*Figure 38:* For indirect use, the recommended widget is Hildon Touch List Picker with multiple selection.

- Spinners (`GtkSpinButton`)

*Figure 39:* For "embedded use", the recommended widget is Hildon Touch List in single selection mode.

Figure 40: For indirect use, the recommended widget is Hildon Touch List Picker.

Figure 41: For very high amounts of values (more than 200 items) or unlimited values, use only `HildonEntry`.

## 19.1 Quick Reference: Used / Deprecated

### 19.1.1 Hildon Widgets

Many of the widgets in Hildon 2.1 are deprecated in Hildon 2.2 and replaced with finger-usable variants.

| State | Widget | Description |
|---|---|---|
| LEGACY | HildonCaption | Left-aligned GtkLabel(s) with GtkSizeGroup should be used instead. |
| DEPRECATED | HildonBreadCrumb | REPLACED by Hildon Touch View(s) |
| DEPRECATED | HildonDialog | REPLACED by GtkDialog |
| DEPRECATED | HildonNumberEditor | REPLACED by HildonEntry |
| DEPRECATED | HildonRangeEditor | REPLACED by HildonEntry / two HildonEntries |
| DEPRECATED | HildonWeekdayPicker<br>HildonCalendarPopup<br>HildonDateEditor | REPLACED by HildonDateButton and HildonDateSelector |
| DEPRECATED | HildonTimePicker<br>HildonTimeEditor | REPLACED by HildonTimeButton and HildonTimeSelector |
| DEPRECATED | HildonControlbar<br>HildonSeekbar<br>HildonVolumebar | REPLACED by GtkV/HScale |
| DEPRECATED | HildonColorButton<br>HildonColorChooserDialog<br>HildonColorChooser | REPLACED by HildonPickerButton and HildonPickerDialog. Build the colour picker using a generic **Hildon Touch List Picker**. |
| DEPRECATED | HildonFontSelectionDialog | REPLACED by HildonPickerButton and HildonPickerDialog. Build the font picker (device has only 4 fonts!) using a generic **Hildon Touch List Picker**. |
| DEPRECATED | HildonCodeDialog | REPLACED by GtkDialog. Applications should create their own dialogue with GtkDialog. |
| DEPRECATED | HildonSortDialog | REPLACED by HildonPickerDialog. Use the basic HildonPickerDialog (without dialogue buttons) to list the sort choices as a single list. |
| DEPRECATED | HildonLoginDialog<br>HildonGetPasswordDialog<br>HildonSetPasswordDialog | REPLACED by GtkDialog. Applications should create their own dialogue with GtkDialog. |

Table 26: Hildon widgets

## 19.2 Gtk Widgets

There are many, many Gtk-widgets which are not used in Maemo at all. See the No change list for suggestions about that. The list below contains widgets that are still in use.

| State | Widget | Description |
|---|---|---|
| HILDON 2.2 | GtkLabel<br>GtkImage<br>GtkProgressBar<br>GtkHScale<br>GtkVScale<br>GtkTreeView<br>GtkIconView | Used normally in both legacy and Hildon 2.2 apps. |
| LEGACY | GtkEntry | REPLACED by HildonEntry |
| LEGACY | GtkTextView | REPLACED by HildonTextView |
| LEGACY | GtkExpander | REPLACED by HildonPannableArea. Any show/hide toggling is not recommended for Hildon 2.2. |
| LEGACY | GtkNotebook | REPLACED by HildonPannableArea. Tabs of any kind are NOT used in Hildon 2.2 |
| LEGACY | GtkScrolledWindow | REPLACED by HildonPannableArea. No scrollbars, direct finger panning instead. |
| OBSOLETE | GtkMenuBar | REPLACED by HildonAppMenu |
| OBSOLETE | GtkColorSelectionDialog<br>GtkFontSelectionDialog<br>GtkFontButton | REPLACED by HildonPickerButton and HildonPickerDialog. Build the colour/font picker using generic **Hildon Touch List Picker**. |
| OBSOLETE | GtkFileChooserButton | Use widgets defined in **hildon-fm (Hildon file management dialogs)** |
| OBSOLETE | GtkStatusBar | Use the content area instead, there is no equivalent UI element to the status bar in Hildon 2.2 |
| OBSOLETE | GtkPageSetupUnixDialog<br>GtkPrintUnixDialog | PRINTING UI NOT SUPPORTED |

Table 27: Gtk widgets